

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ ” _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для аналізу даних при купівлі-продажу
криптовалют на ринку бірж

Виконав студент IV курсу, групи ІТ-84в
(шифр групи)

Соловей Максим Романович
(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник доцент, к.т.н., Лісовиченко О. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант
з графічної
документації ст. вик., Вітковська І. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
комп'ютеризованих систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Солов'ю Максиму Романовичу

(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для аналізу даних при купівлі-продажу криптовалют на ринку бірж

керівник проєкту Лісовиченко Олег Іванович, доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « »квітня 2023 р. №

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Аналіз предметної області: опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог.

2) Проектування крипто-парсера: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення

3) Розробка програмного забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення.

4) Тестування системи: опис особливостей тестування.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань. _____

2) Схема структурна станів системи _____

3) Схема структурна класів програмного забезпечення _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв'язання задачі	01.04.2023	
3	Постановка та формалізація задачі	05.04.2023	
4	Розробка інформаційного забезпечення	15.04.2023	
5	Алгоритмізація задачі	20.04.2023	
6	Обґрунтування вибору використаних технічних засобів	01.05.2023	
7	Розробка програмного забезпечення	15.05.2023	
8	Налагодження програми	20.05.2023	
9	Виконання графічних документів	26.05.2023	
10	Оформлення пояснювальної записки	02.05.2023	
11	Подання ДП на попередній захист	06.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент

(підпис)

Максим СОЛОВЕЙ

(ініціали, прізвище)

Керівник

(підпис)

Олег ЛІСОВИЧЕНКО

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 19 таблиць, 4 рисунків та 10 джерел – загалом 65 сторінок.

Дипломний проєкт присвячений темі : Програмне забезпечення для аналізу даних при купівлі-продажу криптовалют на ринку бірж.

Метою проєкту є підвищення швидкості та надійності подачі корисної інформації для трейдингу (за допомогою телеграм боту).

Об'єкт дослідження: програмне забезпечення для трейдингу.

Предмет дослідження: процеси розроблення, забезпечення якості, впровадження і супроводження програмного забезпечення.

У першому розділі формалізовано задачу побудови крипто-парсера та розглянуто класичні методи для вирішення цієї задачі. Описано недоліки та переваги цих методів.

Другий розділ присвячено дослідженням специфіки предметної області та опису існуючих рішень. Також у цьому розділі проведено обґрунтування обраного методу для рішення поставленої задачі та спроектовано систему.

Розділ три присвячено питанням розробки програмного забезпечення. Цей розділ має містити опис компонентів та класів програмного забезпечення.

В останньому четвертому розділі, розглянути питання тестування системи та валідації отриманих результатів.

КЛЮЧОВІ СЛОВА: КРИПТОВАЛЮТА, ПАРСЕР, API, Біржі, телеграм-бот.

ABSTRACT

The explanatory note of the diploma project consists of four chapters, contains 19 tables, 4 figures and 10 sources - a total of 60 pages.

The diploma project is dedicated to the topic: Software for data analysis when buying and selling cryptocurrencies on the exchange market.

The aim of the project is to increase the speed and reliability of providing useful information for trading (using a telegram bot).

Research object: trading software.

Subject of research: processes of software development, quality assurance, implementation and maintenance.

The first section formalises the problem of building a crypto parser and considers classical methods for solving this problem. The disadvantages and advantages of these methods are described.

The second section is devoted to the study of the specifics of the subject area and a description of existing solutions. This section also justifies the chosen method for solving the problem and designs the system.

Chapter Three is devoted to software development. This section should include a description of software components and classes.

In the last chapter, the fourth section deals with testing the system and validating the results.

KEYWORDS: CRYPTOCURRENCY, PARSER, API, EXCHANGES, TELEGRAM-BOT.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для аналізу даних при купівлі-продажу
криптовалют на ринку бірж
Технічне завдання
КПІ.ІТ-8426.045430.01.81**

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Максим СОЛОВЕЙ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2 ПІДСТАВА ДЛЯ РОЗРОБКИ	4
3 ПРИЗНАЧЕННЯ РОЗРОБКИ	5
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
4.1 Вимоги до функціональних характеристик.....	6
4.2 Умови експлуатації	6
4.3 Вимоги до складу і параметрів технічних засобів.....	7
4.4 Вимоги до маркування та пакування	7
4.5 Вимоги до транспортування та зберігання.....	7
4.6 Спеціальні вимоги.....	7
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ.....	8
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ	9
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ.....	10

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для аналізу даних при купівлі-продажу криптовалют на ринку бірж.

Галузь застосування: торгівля на біржах криптовалют.

Наведене технічне завдання поширюється на розробку телеграм боту, котрий надає список бірж з вигідним курсом на поточний час, з метою підвищення швидкості та надійності подачі корисної інформації для трейдингу.

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки парсера криптовалют для надання корисної інформації є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання інформації про вигідну різницю курсу, для арбітражу, з актуальними біржами на поточний час.

Метою розробки є підвищення швидкості та надійності подачі корисної інформації для трейдингу.

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1.1 Для користувача:

– Програмне забезпечення повинно мати зручний інтерфейс, який дозволяє користувачам легко взаємодіяти з синтаксичним аналізатором, вводити дані та переглядати результати обробки.

4.1.2 Розробку виконати на платформі Windows 11

4.1.3 Додаткові вимоги:

– Інтеграція та доступність. Інтеграція з існуючими інструментами і платформами має вирішальне значення для безперебійного аналізу даних.

4.1.4 Передбачити контроль введення інформації.

4.1.5 Передбачити стандарт для зберігання оброблених даних та версій моделей.

4.2 Умови експлуатації

4.2.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.2.2 Обслуговування

– Зміна вибірки бірж та валют, оновлення програмного забезпечення.

4.2.3 Обслуговуючий персонал

– В ролі обслуговуючого персоналу є надавач користуванням програмного забезпечення. У разі необізнаності користувача знанням мови

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

розробки та середовища. Надавач, має вносити зміни, згідно бажань користувача.

4.3 Вимоги до складу і параметрів технічних засобів.

4.3.1 Програмне забезпечення повинно не має бути залежною від програмної платформи, а отже функціонувати на серверах та комп'ютерах з операційною системою Windows/Ubuntu/MacOS.

4.3.2 Мінімальна конфігурація технічних засобів:

4.3.2.1 Тип процесору Inrel Core i3.

4.3.2.2 Об'єм ОЗП 8 GB.

4.4 Вимоги до інформаційної та програмної сумісності

– Успішний крипто-парсер повинен надавати API або інтеграцію з популярним програмним забезпеченням, торговими платформами або інструментами аналізу даних. Він також повинен бути легкодоступним, дозволяючи користувачам отримувати доступ до даних і аналізувати їх за допомогою зручних інтерфейсів або API.

4.4.1 Вхідні дані повинні бути представлені в наступному форматі:

– API.

4.4.2 Результати повинні бути представлені в наступному форматі:

– Повідомлення.

4.4.3 Програмне забезпечення повинно реалізовувати моделі порівняння.

4.4.4 Програма має бути реалізована мовою програмування Python 3.

4.5 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред'являються.

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

4.6 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.7 Спеціальні вимоги

Не висуваються.

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		8

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування

5.4 Графічна частина повинна бути виконана не менше ніж на 3 аркушах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структура інформаційної системи.

5.4.2 Схема структура потоків даних програмного забезпечення або його частини.

5.4.3 Схема структурна класів програмної системи.

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Таблиця 6.1 – Стадії і етапи розробки

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03.2023	
2.	Аналіз існуючих методів розв'язання задачі	01.04.2023	
3.	Постановка та формалізація задачі	05.04.2023	Технічне завдання
4.	Розробка інформаційного забезпечення	15.04.2023	Схема структурна програмного забезпечення
5.	Алгоритмізація задачі	20.04.2023	
6.	Обґрунтування вибору використаних технічних засобів	02.05.2023	
7.	Розробка програмного забезпечення	15.05.2023	Тексти програмного забезпечення
8.	Налагодження програми	20.05.2023	Графічний матеріал проекту
9.	Виконання графічних документів	26.05.2023	Графічний матеріал проекту

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

					КПІ.ІТ-8426.045430.01.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		11

Пояснювальна записка до дипломного проєкту

на тему: Програмне забезпечення для аналізу даних при купівлі-продажу
криптовалют на ринку бірж

КПІ.ІТ-8426.045430.02.81

Київ – 2023

ЗМІСТ

ВСТУП	5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	7
1.1. Загальні положення	7
1.2. Аналіз особливостей криптовалютного трейдингу.....	9
1.3. Аналіз існуючих технологій та успішних ІТ-проектів	11
1.4. Аналіз вимог до програмного забезпечення.....	14
1.5. Постановка задачі	21
ВИСНОВКИ ДО РОЗДІЛУ	24
2 ПРОЕКТУВАННЯ КРИПТОПАРСЕРА	25
2.1. Моделювання та аналіз програмного забезпечення.....	25
2.2. Огляд архітектури програмного забезпечення	25
2.1.1. На стороні сервера.....	27
2.1.2. Бібліотека трикутного арбітражу.....	28
2.3. Опис і обґрунтування обраного методу	30
2.4. Специфікація вимог до програмного забезпечення	32
ВИСНОВКИ ДО РОЗДІЛУ	36
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	38
3.1. Засоби розробки	38
3.2. Архітектура програмного забезпечення.....	40
3.2.1. Опис компонентів.....	40
3.2.2. Опис класів ПЗ.....	43
3.3. Розгортання та запуск ПЗ.....	49
3.4. Варіанти використання	50
ВИСНОВКИ ДО РОЗДІЛУ	52
4 ТЕСТУВАННЯ СИСТЕМИ	54
4.1. Аналіз якості ПЗ.....	54
4.2. Метрики парсеру криптовалют	54
4.3. Валідація моделі.....	56

ВИСНОВКИ ДО РОЗДІЛУ 58

ВИСНОВКИ..... 59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 60

ДОДАТКИ..... 61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	– Це набір програмного коду, який забезпечує передачу даних між одним програмним продуктом та іншим. Він також містить умови цього обміну даними.
RSI	– Осцилятор імпульсу, який вимірює швидкість і зміну руху ціни.
ICO	– Одна із форм колективного фінансування, краудфандінгу, спосіб залучення фінансових інвестицій у проект.
XML	– Технологія для побудови інтерактивних веб-додатків, яка оброблює запити користувача одразу при їх приході.
CDN	– Розподілена система, що зберігає кешовані корисні файли для надання доступу користувачам по всьому світу.

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

ВСТУП

Криптографія почала набирати оберти з 1990 року, з метою конфіденційних платежів в системі DigiCash, проте ця платіжна система була централізованою і збанкрутувала в 1998 році.

З моменту появи першої децентралізованої крипто валюти Bitcoin, яка використовувала хешування SHA-256 і систему proof of work та мала відкрите програмне забезпечення, почали з'являтися різноманітні проекти. До складу цих проектів входять : нові крипто валюти (Ethereum, Lite coin та інші), ринки крипто валют, біржі крипто валют (Binance, Kraken, Coinbase Exchange, Huobi, Bitstamp та інші), боти, компанії, засновані на крипто валютах тощо.

Біржі крипто валют привернули багато уваги протягом останніх років. На цих біржах перебуває багато грошей, саме це і зацікавило багатьох людей, адже там є можливість отримати вигоду. Крипто валютні біржі не схожі на ринок валют, існує фактор валютної неефективності, з якої можна отримати вигоду.

Ринок крипто валют залежить від багатьох чинників, на його стан можуть впливати навіть новини, пости в різних популярних соціальних мережах. Окрім арбітражу, ефекту календаря, існують економічні бульбашки, котрі спричинені різними стимулами. Ще однією можливістю заробітку, є трекінг декількох обмінних курсів, на різних крипто валютних біржах.

Для того, щоб трейдеру мати стабільний позитивний прибуток, при купівлі-продажу крипто валюти, інвесторам, та трейдерам необхідно мати інформацію з багатьох бірж. Вчасне бачення ринку крипто валют на багатьох біржах, дає змогу швидше зорієнтуватись та не впустити момент вдалої купівлі або продажу крипто валюти.

Людство має зацікавленість в корисних та незамінних речах. Корисна річ має ціну і попит. Первісна цінність крипто валюти – це вартість витраченої електроенергії. Вторинною вартістю є чинники які визначають попит

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

на цю крипто валюту. Попит може бути : спекулятивно-інвесторський, купівля товару (за крипто валюту), переказом на інший рахунок без комісії.

Велика кількість трейдерів почала цікавитись інформацією, як спростити моніторинг ринків крипто валют, що призвело до пошуку рішення цього питання. Рішенням якого може бути парсинг ринку крипто валют за допомогою API бірж.

Метою цієї роботи є підвищення швидкості та надійності подачі корисної інформації для трейдингу. Створення персонального боту в месенджері Telegram, за допомогою якого можна слідкувати за ринком крипто валют, правильно та вчасно робити транзакції на різних біржах з різними криптовалютами.

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальні положення

Парсер - це інструмент, розроблений постачальником, який дозволяє користувачам отримувати дані та інформацію з різних криптовалютних бірж для аналізу та досліджень. Парсер може збирати, аналізувати та представляти дані з різних бірж, залежно від доступності та функціональності API бірж.

Загальні положення та умови регулюють ваш доступ і використання парсеру криптовалют, що надається постачальником. Отримуючи доступ до парсеру або використовуючи його, ви погоджуєтесь дотримуватися цих умов. Якщо ви не згодні з цими умовами, ви не можете використовувати Парсер.

Розробник гарантує, що парсер не матиме дефектів матеріалів та виготовлення протягом вказаного гарантійного терміну, з дати поставки.

Розробник надаватиме необхідну підтримку та допомогу протягом гарантійного періоду для усунення будь-яких дефектів або проблем, виявлених Клієнтом.

Підтримка після закінчення гарантійного терміну може надаватися за окремими умовами та за окрему плату.

Розробник не несе відповідальності за будь-які прямі, непрямі, випадкові або побічні збитки, що виникають внаслідок або у зв'язку з використанням або неможливістю використання парсеру.

Максимальна відповідальність розробника за договором, деліктом або іншим чином не повинна перевищувати загальну суму гонорарів, сплачених Клієнтом за цією Угодою.

Парсер криптовалют, також відомий як парсер криптовалют або парсер блокчейнів, - це програмний інструмент, який використовується для вилучення та аналізу даних з мереж блокчейнів. Це важливий компонент у

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

створенні додатків, які взаємодіють з криптовалютами. Ось огляд основних методів, які використовуються при створенні криптопарсеру:

- Підключення до мережі блокчейн. Першим кроком у створенні криптоаналізатора є встановлення з'єднання з мережею блокчейн цільової криптовалюти. Це можна зробити за допомогою мережевих API (інтерфейсів прикладного програмування) або шляхом запуску повноцінного вузла для безпосередньої взаємодії з блокчейном. API забезпечують більш зручний та ефективний спосіб отримання даних блокчейну.

- Зберігання даних. Проаналізовані дані блокчейну потрібно зберігати в базі даних або іншій відповідній системі зберігання даних. Це дозволяє ефективно запитувати і отримувати інформацію для подальшого аналізу.

- Індекссування та запити. Для швидкого та ефективного пошуку даних у блокчейні використовуються методи індексування. Це передбачає створення індексів на відповідних полях, таких як хеші транзакцій, адреси або висота блоків. Індекссування дозволяє швидше знаходити певні елементи даних і підвищує загальну продуктивність. Механізми запитів, такі як SQL або користувацькі мови запитів, можуть бути реалізовані для отримання певних підмножин даних на основі різних критеріїв.

- Аналітика та візуалізація. Важливим аспектом крипто-парсера, є надання інструментів для аналізу та візуалізації даних. Це передбачає реалізацію алгоритмів і методів для отримання інформації з аналізованих даних блокчейну. Наприклад, аналіз шаблонів транзакцій, ідентифікація адрес, що належать одному суб'єкту, виявлення аномалій або генерація статистики. Візуалізація даних за допомогою діаграм, графіків або інформаційних панелей допомагає краще зрозуміти та інтерпретувати інформацію.

- Оновлення та обслуговування. Криптовалюти постійно розвиваються, оновлюються протоколи, змінюються мережі та впроваджуються нові функції. Парсер криптовалют повинен бути розроблений таким чином,

щоб обробляти ці оновлення і забезпечувати сумісність з останніми версіями мережі блокчейн. Регулярне технічне обслуговування та оновлення необхідні для забезпечення функціональності та надійності парсеру.

Створення криптоаналізатора вимагає хорошого розуміння технологій блокчейну, структур даних і навичок програмування. Він включає в себе поєднання методів вилучення, аналізу, зберігання, індексування, запитів, аналітики та візуалізації даних для отримання цінної інформації з блокчейн-даних.

1.2. Аналіз особливостей криптовалютного трейдингу

Попит на ринку крипто валют обумовлений інтересом у вкладі в інвестиції, можливістю вигідних транзакцій валюти з будь-якого місця та спекулятивних чинниках.

Задачею корисних додатків для трейдингу на ринку бірж – вчасно постачати важливу та необхідну інформацію, нею можуть бути:

- курс валют;
- стан різних бірж;
- прогнозування зросту чи спадання, тої чи іншої валюти;
- супроводжувати угоди;
- накопичувати чи обмінювати валюту;
- тощо.

Для гарантування надання коректної інформації, телеграм ботом, треба забезпечити збір даних в режимі реального часу, та переконатися у виборі надійних джерел.

Інтернет-трейдинг – має на увазі торгівлю на фінансових ринках за допомогою зручних онлайн бірж (Coinbase Pro, Coinmate, Bittrex, Bitstamp, Bitfinex, Binance, Kraken, Huobi, OKEx, DigiFinex та інші). Значною перевагою гри, купівлі-продажу, є низька вартість обслуговування операцій для трейдерів.

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		9

Криптоаналізатори, також відомі як парсери криптовалют, - це програмні інструменти або алгоритми, призначені для вилучення та аналізу даних з мереж блокчейн і криптовалютних транзакцій. Успішні криптоаналізатори мають кілька ключових характеристик, які сприяють їхній ефективності.

Швидкість та ефективність. Ринок криптовалют працює в режимі реального часу, і своєчасний аналіз даних має вирішальне значення для прийняття обґрунтованих рішень. Успішний криптопарсер повинен бути швидким і ефективним в обробці та аналізі даних.

Точність і надійність. Точність даних є життєво важливою при аналізі криптовалют. Успішний криптоаналізатор повинен надавати точну інформацію, включаючи суми транзакцій, часові мітки, адреси та інші важливі деталі. Парсер повинен покладатися на надійні алгоритми і методи перевірки даних, щоб забезпечити цілісність видобутих даних.

Безпека і конфіденційність. Криптовалюти пов'язані з конфіденційними фінансовими даними, тому безпека має першорядне значення. Успішний криптоаналізатор повинен надавати пріоритет заходам безпеки для захисту даних користувача, приватних ключів та іншої конфіденційної інформації. Він повинен використовувати шифрування, безпечні протоколи передачі даних і дотримуватися найкращих практик кібербезпеки.

Гнучкість і кастомізація. Різні користувачі мають різні потреби в аналізі криптовалютних даних. Успішний криптоаналізатор повинен пропонувати гнучкість і можливості кастомізації, щоб задовольнити різні вимоги. Це включає можливість вибирати певні точки даних для аналізу, визначати кастомні фільтри і генерувати звіти відповідно до індивідуальних уподобань.

Інтеграція та доступність. Інтеграція з існуючими інструментами і платформами має вирішальне значення для безперебійного аналізу даних. Успішний крипто-парсер повинен надавати API або інтеграцію з

популярним програмним забезпеченням, торговими платформами або інструментами аналізу даних. Він також повинен бути легкодоступним, дозволяючи користувачам отримувати доступ до даних і аналізувати їх за допомогою зручних інтерфейсів або API.

Постійний розвиток та оновлення. Криптовалютний ландшафт швидко розвивається, регулярно з'являються нові протоколи блокчейну, криптовалют та динаміка ринку. Успішний криптоаналізатор повинен мати спеціальну команду розробників, яка буде в курсі останніх подій і регулярно оновлюватиме його для забезпечення сумісності та ефективності.

Документація та підтримка. Вичерпна документація та надійні канали підтримки необхідні для того, щоб користувачі могли зрозуміти та максимально використати можливості криптоаналізатора. Успішний парсер повинен надавати чітку документацію, посібники користувача, навчальні матеріали та оперативну підтримку, щоб допомогти користувачам ефективно використовувати інструмент.

Варто зазначити, що успіх криптоаналізатора в кінцевому рахунку залежить від того, наскільки добре він відповідає конкретним потребам і вимогам своїх користувачів. Різні люди та організації можуть надавати пріоритет певним факторам над іншими, залежно від їхніх випадків використання та цілей.

1.3. Аналіз існуючих технологій та успішних IT-проектів

Основною технологією, що лежить в роботі Bitcoin, є блокчейн. Він має публічний код і веде запис усіх транзакцій, котрі трапились в платіжній системі. Блокчейн побудований з блоків, які поєднані таким чином, що один блок містить хеш заголовка попереднього блоку, а попередній заголовок наступного хеш блоку. Ця схема може бути уявленою як лінійна зв'язка послідовних блоків. Перший блок, генезисний, був створений у 2009 році. Ці

блоки мають файли зі збереженням даних транзакцій, їх можна розділити на наступні частини :

- розмірність блоку (кількість байтів після кінця блоку);
- заголовок блоку (складається з інших шести елементів);
- магічне число (значення якого завжди складає 0xD9B4BEF9, 4 байти);
- лічильник транзакцій (завжди додатне ціле число);
- транзакції (не порожній список інформації про здійснені переводи).

Транзакції складаються з кількох елементів, таких як: яка кількість грошового еквіваленту відправлена, з якого рахунку в який відбувається транзакція, а також, невід’ємною частиною є свідки, котрі забезпечують підписами для необхідної перевірки транзакцій. Ці данні були відокремлені від блоку ще в 2017 році, через проблему, пов’язану з труднощами в мережі Bitcoin.

Крім аналізу ринку крипто валют, крипто валютних бірж та трейдерів, також оглядаються вже існуючі рішення. Всі вони зав’язані з перехресним обміном, звичайним та тріангулярним арбітражем.

Існує безліч простих програм для арбітражу, як веб додатки так і настільні програми:

Одним із добре зарекомендованих веб додатків, є Crypto Arbitrage Opportunity. Він надає поточні вигідні арбітражні можливості у відсотках, серед великої кількості бірж криптовалют. Надає вибір перегляду різних бірж криптовалют, та торгові пари. Приклад надання інформації сайтом, продемонстровано на рисунку 1.1.



Рисунок 1.1 – Crypto Arbitrage Opportunity, сайт з надання корисних торгових пар.

Сайт надає платні послуги, серед яких, в головному меню можна вибрати арбітражні розповсюдження та арбітражні шляхи.

Існують також і настільні програми, які є менш організованими та популярними серед трейдерів. Гарними прикладами є програми: Arbitrage Opportunities for Cryptocurrencies та Haas Online. Вони надають різні пропозиції, згідно з тарифним планом, в залежності від доступних функцій.

На GitHub, знаходиться безліч проектів для зручного встановлення та користування. Гарними прикладами є програми : Blackbird, представлений за ліцензією Масачусетського Інституту Технологій. Використовує автоматичний арбітраж на багатьох відомих біржах криптовалют. Написаний та запрограмований на мові C.

Різновидом настільних програм є – триангулярні арбітражні програми. Більшість з них, погано задокументовані, не мають графічного інтерфейсу та можуть не отримувати підтримки зі сторони розробника. Прикладом такого різновиду є програмне забезпечення Haas Online. За допомогою нього, користувач може вибрати торгові пари, з якими відбуватиметься взаємодія у триангулярному арбітражі, еквівалент певної валюти та розрахований прибуток, приклад роботи зображено на рисунку 1.2.

Top Potential Arbitrage Triplets, via: BNB,ETH,USDT							
Step A	Step B	Step C	Rate	Fees BnB	(Rate - BnB Fee)	Fees Normal	(Rate - Fee)
BTC	USDT	HOT	1.423%	0.071%	1.352%	0.142%	1.281%
BTC	ETH	ARK	1.099%	0.055%	1.044%	0.110%	0.989%
BTC	ETH	QSP	1.041%	0.052%	0.989%	0.104%	0.937%
BTC	BNB	AMB	1.041%	0.052%	0.989%	0.104%	0.937%
BTC	BNB	GTO	1.017%	0.051%	0.967%	0.102%	0.916%
BTC	BNB	DLT	0.976%	0.049%	0.927%	0.098%	0.879%
BTC	ETH	POA	0.896%	0.045%	0.852%	0.090%	0.807%
BTC	ETH	ARN	0.809%	0.040%	0.769%	0.081%	0.728%
BTC	ETH	OST	0.786%	0.039%	0.747%	0.079%	0.707%
BTC	BNB	ZIL	0.750%	0.037%	0.712%	0.075%	0.675%
BTC	ETH	VET	0.708%	0.035%	0.672%	0.071%	0.637%
BTC	USDT	ZIL	0.698%	0.035%	0.663%	0.070%	0.628%
BTC	ETH	WABI	0.675%	0.034%	0.641%	0.068%	0.608%
BTC	USDT	OMG	0.555%	0.028%	0.527%	0.055%	0.499%
BTC	ETH	DLT	0.539%	0.027%	0.512%	0.054%	0.485%
BTC	ETH	ZIL	0.532%	0.027%	0.505%	0.053%	0.479%
BTC	BNB	DASH	0.517%	0.026%	0.491%	0.052%	0.465%
BTC	ETH	BTG	0.514%	0.026%	0.488%	0.051%	0.463%
BTC	BNB	QLC	0.457%	0.023%	0.434%	0.046%	0.411%
BTC	BNB	WTC	0.437%	0.022%	0.415%	0.044%	0.393%
BTC	BNB	OMG	0.423%	0.021%	0.402%	0.042%	0.381%

Рисунок 1.2 – Приклад триангулярної арбітражної програми.

Для прикладу складно знайти програму з відкритим вихідним кодом, яка задовольняє більшість трейдерів, та показує можливості триангулярного арбітражу в режимі реального часу та ще й на кількох біржах криптовалют.

1.4. Аналіз вимог до програмного забезпечення

Метою цього документу є аналіз вимог до програмного забезпечення криптоаналізатора. Програмне забезпечення для аналізу криптографічних даних призначене для аналізу та інтерпретації криптографічних даних, таких як зашифровані повідомлення, цифрові підписи та криптографічні протоколи. Воно відіграє вирішальну роль у забезпеченні безпеки та цілісності даних у різних сферах, включаючи фінанси, комунікації та інформаційну безпеку.

Таблиця 1.1 — Варіант використання UC-1

Use Case Name	Переглянути дані криптовалюти
Use case ID	UC-1
Goals	Перегляд даних криптовалют
Actors	Користувач
Trigger	Користувач бажає переглянути ринок валют
Pre-conditions	-
Flow of Events	Перехід за посиланням
Extension	-
Post-Condition	Перегляд даних криптовалют

Таблиця 1.2 — Варіант використання UC-2

Use Case Name	Переглянути метрики криптовалютного ринку
Use case ID	UC-2
Goals	Перегляд метрик криптовалютного ринку
Actors	Користувач
Trigger	Користувач бажає переглянути метрики криптовалютного ринку
Pre-conditions	-
Flow of Events	Користувачу надається інформація, за запитом
Extension	-
Post-Condition	Перегляд метрик криптовалютного ринку

Таблиця 1.3 — Варіант використання UC-3

Use Case Name	Переглянути останні новини про криптовалюту
Use case ID	UC-3
Goals	Перегляд останніх новин про криптовалюту
Actors	Користувач
Trigger	Користувач бажає переглянути останні новини про криптовалюту
Pre-conditions	-
Flow of Events	Перехід за посиланням
Extension	-
Post-Condition	Перегляд останніх новин про криптовалюту

Таблиця 1.4 — Варіант використання UC-4

Use Case Name	Оновити історичні дані системи
Use case ID	UC-4
Goals	Оновлення історичних даних системи
Actors	Адміністратор
Trigger	Адміністратор бажає оновлення історичних даних системи
Pre-conditions	-
Flow of Events	Оновлення даних, запуском програмного забезпечення
Extension	-
Post-Condition	Оновлення історичних даних системи

Таблиця 1.5 — Варіант використання UC-5

Use Case Name	Зареєструвати особу
Use case ID	UC-5
Goals	Долучити нового користувача
Actors	Адміністратор
Trigger	Адміністратору дано завдання долучити нового користувача
Pre-conditions	-
Flow of Events	Адміністратор долучає нового користувача у телеграм-канял
Extension	-
Post-Condition	Зареєстровано нового користувача

Таблиця 1.6 — Варіант використання UC-6

Use Case Name	Створити запит до API
Use case ID	UC-6
Goals	Створення запиту до API
Actors	Адміністратор
Trigger	Адміністратор бажає створити запит до API
Pre-conditions	-
Flow of Events	Створення запитів до бірж та валют, зацікавлених користувачем
Extension	Якщо серед обраних бірж не має цільової інформації, програмне забезпечення чекатиме рішення
Post-Condition	Надання даних, в котрих зацікавлений користувач

Таблиця 1.7 – Функціональна вимога FR-1

Назва	Синтаксичний аналіз та декодування
Опис	Програмне забезпечення повинно вміти аналізувати та декодувати вхідні дані, витягуючи відповідну інформацію із зашифрованих повідомлень, сертифікатів.

Таблиця 1.8 – Функціональна вимога FR -2

Назва	Обробка помилок
Опис	Програмне забезпечення повинно коректно обробляти помилки, надаючи змістовні повідомлення про помилки та відповідні дії для усунення або повідомлення про помилки, що виникають під час синтаксичного аналізу або обробки.

Таблиця 1.9 – Функціональна вимога FR -3

Назва	Безпека
Опис	Програмне забезпечення повинно відповідати найвищим стандартам безпеки, забезпечуючи конфіденційність і цілісність криптографічних даних протягом усього процесу розбору. Воно повинно захищати від потенційних вразливостей, включаючи атаки побічних каналів, ін'єкційні атаки та несанкціонований доступ.

Таблиця 1.10 – Функціональна вимога FR -4

Назва	Продуктивність
Опис	Програмне забезпечення має бути ефективним і здатним обробляти великі обсяги криптографічних даних за прийнятний час. Воно повинно оптимізувати алгоритми розбору та обробки для мінімізації обчислювальних витрат.

Таблиця 1.11 – Функціональна вимога FR -5

Назва	Масштабованість
Опис	Програмне забезпечення має бути спроектоване таким чином, щоб обробляти зростаючі обсяги криптографічних даних у міру зростання системи. Воно повинно підтримувати горизонтальне масштабування, розподіляючи робоче навантаження на декілька серверів або процесорів.

Таблиця 1.12 – Функціональна вимога FR -6

Назва	Інтерфейс користувача
Опис	Програмне забезпечення повинно мати зручний інтерфейс, який дозволяє користувачам легко взаємодіяти з синтаксичним аналізатором, вводити дані та переглядати результати аналізу. Програма повинна підтримувати відповідні функції, такі як пошук, фільтрація та сортування даних, що аналізуються.

Таблиця 1.13 – Функціональна вимога FR -7

Назва	Сумісність
Опис	Програмне забезпечення має бути сумісним з різними операційними системами та платформами, що гарантує його безперебійну роботу в різних середовищах.

Таблиця 1.14 – Функціональна вимога FR -8

Назва	Розширюваність
Опис	Програмне забезпечення повинно бути розроблене з урахуванням майбутніх удосконалень та розширень, що дозволить включити нові криптографічні алгоритми або формати.

Таблиця 1.15 – Матриця трасування вимог і варіантів використання

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8
UC-1								
UC-2								
UC-3								
UC-4								
UC-5								
UC-6								

Таблиця 1.16 – Модель вимог

Номер	Назва	Код	Пріоритет	Навантаження	Ризик
1	Переглянути дані крипто-валюти	REQ001	1	2	Низький
2	Переглянути метрики криптовалютного ринку	REQ002	2	3	Середній
3	Переглянути останні новини про криптовалюти	REQ003	3	2	Низький
4	Оновити історичні дані системи	REQ004	4	5	Середній
5	Зареєструвати особу	REQ005	5	1	Низький
6	Створити запит до API	REQ006	6	5	Високий

1.5. Постановка задачі

Перш ніж розробляти парсер криптовалют, слід вирішити кілька завдань. Ось список кроків, які допоможуть розпочати роботу:

- Визначити сферу застосування та вимог. Визначит конкретне призначення і цілі вашого криптоаналізатора. Зрозуміти, які види

криптовалют і мереж блокчейн ви хочете підтримувати, а також які конкретні дані ви хочете витягувати і аналізувати.

- Дослідити криптоекосистему. Отримати глибоке розуміння криптоіндустрії, включаючи різні криптовалюти, технології блокчейн, структури даних і протоколи. Будьти в курсі останніх подій та тенденцій у цій галузі.

- Вибрати мову програмування. Вибрати мову програмування, яка підходить для нашого проекту. Врахувати такі фактори, як продуктивність, підтримка спільноти, доступні бібліотеки та досвід вашої команди.

- Вибрати джерело даних. Визначити джерело(-а) даних, які будуть використовувати для видобування криптографічних даних. Це можуть бути загальнодоступні API блокчейну, біржі або інші відповідні джерела. Переконайтеся, що вибране джерело даних надає необхідні дані у форматі, який можна аналізувати.

- Створити модель даних: Визначити структуру і формат, в якому ви будете зберігати проаналізовані криптографічні дані. Визначитись з полями даних, зв'язками та будь-якими додатковими метаданими, які потрібно зібрати.

- Сплануйте логіку розбору: Розробіть логіку та алгоритми для аналізу криптографічних даних з обраного джерела. Врахуйте формати даних (наприклад, JSON, XML, CSV) і будь-які необхідні перетворення або обчислення.

- Розробити код для вилучення даних: Реалізуйте код для пошуку та вилучення відповідних криптографічних даних з обраного джерела даних. Обробіть всі необхідні механізми автентифікації, обмеження швидкості або пагінації.

- Реалізувати логіку парсингу. Створити код для аналізу витягнутих даних на основі попередньо визначеної логіки та алгоритмів. Перевірка

та очищення даних за необхідності для забезпечення точності та узгодженості.

- Реалізувати обробку помилок і ведення журналу: Включити механізми обробки помилок, щоб ефективно вирішувати будь-які проблеми, які можуть виникнути в процесі синтаксичного аналізу. Реалізувати логування, щоб збирати відповідну інформацію для налагодження та моніторингу.

- Тестування та валідація. Провести ретельне тестування, щоб переконатися, що синтаксичний аналізатор працює коректно і обробляє різні сценарії та граничні випадки. Порівняти отримані дані з очікуваними результатами та перевірте їхню точність.

- Оптимізація продуктивності. Налаштувати продуктивність синтаксичного аналізатора, оптимізуючи код, покращуючи алгоритми та використовуючи кешування або інші методи для мінімізації затримок і споживання ресурсів.

- Задokumentувати парсер. Створити вичерпну документацію, яка пояснює призначення, функціональність, використання та будь-які залежності криптоаналізатора. Надати приклади та фрагменти коду, щоб допомогти користувачам ефективно використовувати синтаксичний аналізатор.

- Породумати безпеку. Оцінити та усунути потенційні ризики безпеки, пов'язані з обробкою криптографічних даних. Вживати необхідних заходів для захисту конфіденційної інформації, таких як шифрування, безпечні протоколи зв'язку та контроль доступу.

- Обслуговування та оновлення. Криптоєкосистеми динамічні та розвиваються, тому треба планувати постійне обслуговування та оновлення вашого парсеру. Бути в курсі змін у джерелах даних, блокчейн-мережах і будь-яких відповідних стандартах, щоб парсер залишався точним і функціональним.

ВИСНОВКИ ДО РОЗДІЛУ

Таким чином, аналіз предметної області надав цінні знання та інформацію, які слугуватимуть основою для розробки парсеру валют. Ретельне вивчення поточних тенденцій, викликів та можливостей на ринку криптовалют допомогло визначити ключові вимоги та міркування для розробки успішної платформи.

Аналіз показав, що ринок криптовалют переживає значне зростання і прийняття, з'являється різноманітне коло користувачів і варіантів використання. Однак це швидке розширення також несе з собою різні виклики, такі як масштабованість, безпека та користувацький досвід. Ці виклики необхідно ретельно враховувати при розробці крипто парсера, щоб забезпечити його ефективність і конкурентоспроможність на ринку.

Крім того, аналіз пролив світло на важливість впровадження передових заходів безпеки та надійних алгоритмів шифрування для захисту активів і транзакцій на платформі. Крім того, зручність інтерфейсу та інтуїтивно зрозумілі функції повинні бути пріоритетними для підвищення доступності та простоти використання, як для досвідчених трейдерів, так і для новачків у криптовалютному просторі.

Крім того, аналіз підкреслив важливість інтероперабельності та інтеграції з існуючими блокчейн-мережами та криптовалютними біржами. Безперешкодна інтеграція із зовнішніми системами дозволить парсеру використовувати переваги вже існуючих мереж, надаючи користувачам всебічний та універсальний досвід торгівлі.

Таким чином, аналіз предметної області заклав основу для розробки парсеру, визначивши потреби, виклики та можливості ринку. Звертаючись до аспектів безпеки, масштабованості, користувацького досвіду та інтеграції, дизайн може бути спрямований на створення передової платформи, яка відповідає вимогам криптовалютного ландшафту, що розвивається.

2 ПРОЕКТУВАННЯ КРИПТОПАРСЕРА

2.1. Моделювання та аналіз програмного забезпечення

Для моделювання архітектури та бізнес-процесів була використана методологія створення діаграм BPMN.

Серед бізнес процесів користування рекомендаційною системи можна виділити наступні:

- вибір чинників;
- збір та аналіз даних;
- формування результату.

На рисунку 2.1 наведено схему розглянутих бізнес-процесів.



Рисунок 2.1 — Діаграма бізнес процесів.

2.2. Огляд архітектури програмного забезпечення

Парсер криптовалют - це програма на мові Python, призначена для збору та аналізу криптовалютних даних. Він надає комплексну архітектуру

для вилучення, обробки та візуалізації інформації з різних джерел криптовалютних даних. Ось огляд його архітектури:

- Отримання даних. Перший компонент парсеру відповідає за отримання криптовалютних даних з різних джерел. Він може підключатися до криптовалютних бірж, блокчейн API або інших постачальників даних для отримання даних в режимі реального часу або історичних даних. Цей компонент виконує пошук даних асинхронно, щоб забезпечити ефективну та паралельну вибірку.

- Обробка даних. Після того, як дані отримані, вони проходять попередню обробку та очищення в компоненті Обробка даних. Цей етап включає перевірку, перетворення та фільтрацію даних. Це гарантує, що дані є точними, узгодженими та готовими для подальшого аналізу. Оброблені дані зазвичай зберігаються у відповідній структурі даних, такій як база даних або фреймворк даних.

- Синтаксичний аналіз та вилучення. Модуль Parser є важливою частиною архітектури, що відповідає за синтаксичний аналіз і вилучення релевантної інформації з отриманих даних. Він використовує різні методи, такі як обробка тексту, регулярні вирази і зіставлення шаблонів, для вилучення конкретних даних, таких як ціни на криптовалюту, обсяги торгів, ринкові тенденції або деталі транзакцій в блокчейні. Потім витягнуті дані структуруються і організовуються для подальшого аналізу.

- Аналіз та обчислення. Витягнуті дані передаються до компонента "Аналіз та обчислення", який виконує низку обчислень і розрахунків. Цей компонент може містити модулі для розрахунку ковзних середніх, волатильності, кореляцій та інших статистичних показників. Він також може реалізовувати технічні індикатори, аналіз настроїв або алгоритми машинного навчання для отримання значущих висновків з даних

- Інтеграція та API. Архітектура парсеру також може включати інтеграцію із зовнішніми сервісами та API. Це дозволяє користувачам

використовувати додаткові функції, такі як доступ до історичних даних від сторонніх постачальників, взаємодія з торговими платформами або підключення до мереж блокчейн.

– Безпека та обробка помилок. Архітектура програмного забезпечення включає положення про безпеку та обробку помилок. Вона забезпечує безпечну обробку конфіденційних даних, реалізує механізми автентифікації та захищає від поширених вразливостей. Механізми обробки помилок створюються для виявлення та обробки винятків, забезпечуючи надійність та стабільність програмного забезпечення.

В цілому, архітектура програмного забезпечення парсеру на мові Python забезпечує гнучкий і модульний фреймворк для розбору, аналізу та візуалізації криптовалютних даних. Це дозволяє користувачам отримувати інформацію про крипторинок, приймати обґрунтовані рішення та розробляти торгові стратегії на основі надійних і точних даних.

2.1.1. На стороні сервера

Серверна частина буде складатися з трьох основних частин - каналів, контролера та моделей. Канали будуть створені таким чином, щоб сервер міг транслювати дані в реальному часі клієнту через компонент Action Cable. Кожна криптовалютна біржа матиме власний канал. Коли користувач обирає іншу криптовалютну біржу буде обрана користувачем, сервер помітить це і зможе відреагувати на цю зміну. Контролер буде обробляти бізнес-логіку, коли буде отримано новий HTTP-запит буде отримано новий HTTP-запит. Зокрема, сторінка історичних даних повинна буде обробляти даними з бази даних. Логіка вибору дати і логіка запиту цієї сторінки також будуть оброблятися в контролері. Коли користувач отримує доступ до сторінки історичних даних, сервер відповість даними того дня, коли було відправлено запит (час UTC), коли було надіслано запит.

Модель буде прямолінійною і досить простою. Оскільки не буде складних взаємозв'язків між завантаженими даними, достатньо буде однієї таблиці в базі даних буде достатньо. Таблиця буде складатися з:

- Exchange name – type :string
- Date – type :datetime
- Profit – type :float
- Currencies – type :string
- Pair #1 – type :string
- Pair #2 – type :string
- Pair #3 – type :string

2.1.2. Бібліотека трикутного арбітражу

Бібліотека трикутного арбітражу буде представляти собою модуль, який буде взаємодіяти з сторонніми API - криптовалютними біржами. У ньому будуть реалізовані методи, які завантажують дані в реальному часі з обраних криптовалютних бірж. Якщо використовується REST API, дані в реальному часі будуть завантажуватися періодично якомога частіше (бажано щосекунди). В іншому випадку WebSockets і в цьому випадку дані в реальному часі будуть надсилатися клієнту, як тільки відбудуться зміни. клієнту в режимі реального часу, як тільки відбудуться зміни. Буде обрано дві криптовалютні біржі, і ця бібліотека буде завантажуватиме з них дані в реальному часі. Найкращою криптовалютною біржею Kraken через те, що пропонується лише REST API, незважаючи на його великий на момент написання цього тексту пропонується лише REST API, незважаючи на його великі обсяги. Тому тут можуть бути хороші можливості для трикутного можливості для трикутного арбітражу, оскільки ліміт для публічних запитів становить 1 запит/сек. і трейдери можуть звертатися до API в різні моменти часу. Крім того, Kraken є найбільшою біржею біткоїнів з огляду на обсяг в євро та ліквідності. Іншою біржею, яка підходить для

цього завдання, є Bittrex, оскільки її REST API пропонує кінцеву точку, яка повертає всі дані, необхідні для нашого завдання. На жаль, опція WebSocket вимагає SignalR, а зараз немає (підтримуваних) клієнтів SignalR для Ruby. Крім того, Bittrex добре відомий своєю безпекою - її ніколи не зламували. Це одна з найбільших криптовалютних бірж найбільших криптовалютних бірж за кількістю торгових пар - вона пропонує більше 300 з них. Крім того, вона користується повагою у всій блокчейн-спільноті. Окрім завантаження даних в режимі реального часу, модуль також обробляє дані з метою отримання потенційного прибутку. Буде створено клас Exchange, який буде суперкласом для всіх реалізованих криптовалютних бірж. Він буде створювати запити, надсилати їх до сторонніх API та отримувати відповіді. Він буде також обчислюватиме потенційний прибуток і готуватиме оброблені дані для відправки клієнту. Потім з'являться KrakenExchange і BittrexExchange, які успадкують від Біржі. Кожен з них буде реалізовувати методи які будуть обробляти дані у відповідь і створювати з них спеціальні структури. Нам знадобляться колекції пар і трикутників. Колекція пар буде зберігати всі пари на даній криптовалютній біржі. Колекція трикутників буде зберігати всі можливі трикутники (наприклад, BTC → LTC → USD) за допомогою яких можна здійснювати трикутний арбітраж. Крім того, згаданий суперклас

суперклас також оголосить методи, які потрібно буде перевизначити у підкласах такі як create_pairs, update_pairs, create_triangles. Загальна картина загальну картину показано на рисунку 2.2 :

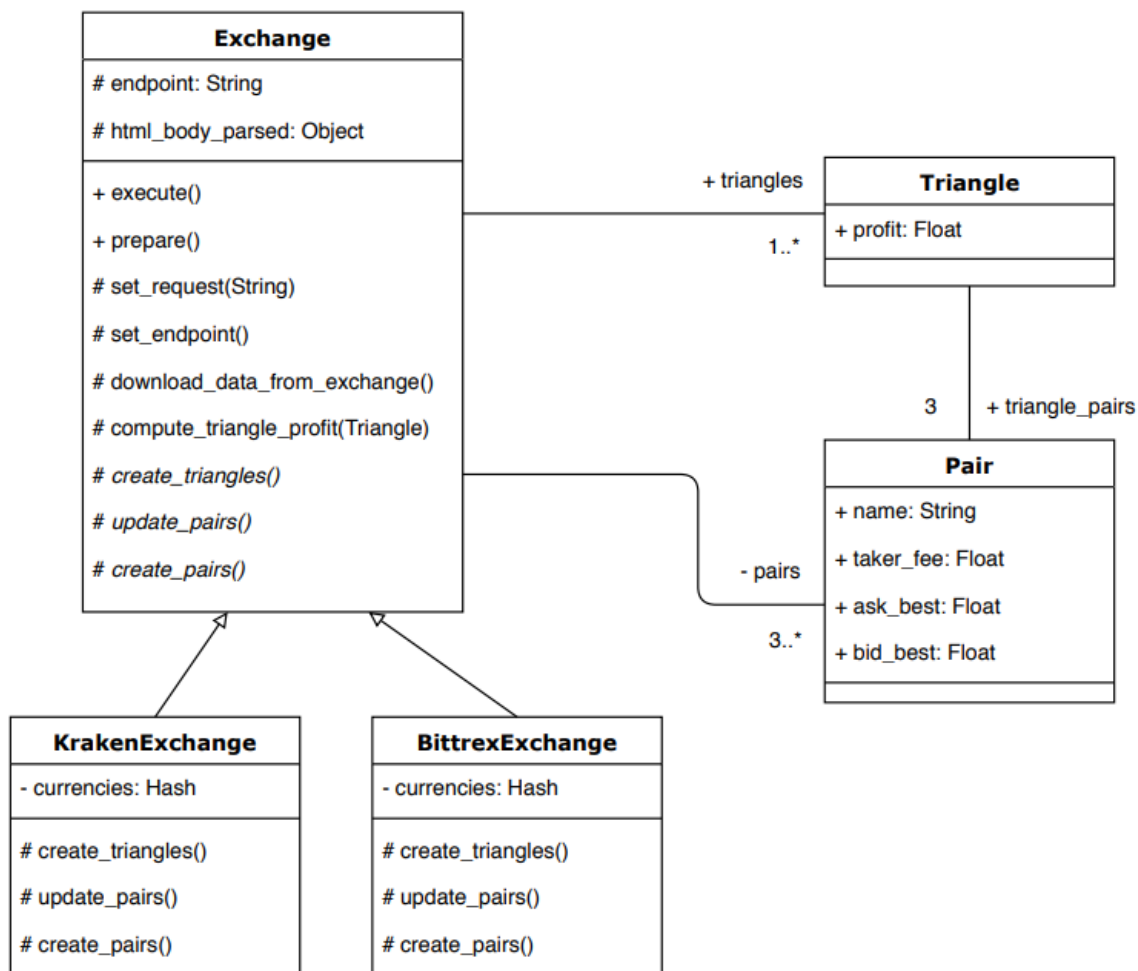


Рисунок 2.2 — Діаграма класів

2.3. Опис і обґрунтування обраного методу

Парсер криптовалют на Python - це програма або модуль, призначений для вилучення та обробки інформації з різних джерел даних, пов'язаних з криптовалютами, таких як криптовалютні біржі, блокчейн-мережі або веб-сайти з криптовалютними новинами. Він призначений для отримання конкретних даних, таких як ціни, обсяги торгів, ринкові тенденції, статті новин або будь-яку іншу інформацію, пов'язану з криптовалютами.

Створення крипто-парсера на мові Python може бути виправдано з кількох причин:

1. Аналіз даних. Криптовалюти відомі своєю волатильністю, і аналіз історичних даних може дати уявлення про ринкові тенденції та

закономірності. Парсер криптовалют може отримувати історичні дані про ціни, обсяги торгів та інші показники, що дозволяє користувачам проводити глибокий аналіз даних і приймати обґрунтовані інвестиційні рішення.

2. Моніторинг в режимі реального часу: Трейдерам та інвесторам потрібна актуальна інформація про ціни на криптовалюту та рух ринку. Парсер криптовалют може безперервно отримувати дані в режимі реального часу з різних джерел, надаючи користувачам миттєвий доступ до останніх ринкових оновлень, забезпечуючи їх постійну обізнаність про постійно мінливий криптовалютний ландшафт.

3. Управління портфелем: Кripto-ентузіасти часто керують різноманітними портфелями, що складаються з різних криптовалют. Кriptoаналізатор може отримувати баланси, історії транзакцій та інші важливі дані з різних криптовалютних бірж, що дозволяє користувачам консолідувати та ефективно управляти своїми портфелями.

4. Агрегація новин: Відстеження новин та оголошень, пов'язаних з криптовалютами, має вирішальне значення для розуміння ринкових настроїв і потенційного впливу на ціни. Кriptoаналізатор може аналізувати новини, публікації в блогах і стрічки соціальних мереж, об'єднуючи інформацію в єдине джерело, щоб користувачі були в курсі останніх подій у криптовалютному просторі.

5. Дослідження та освіта: Розробники, дослідники та викладачі часто потребують доступу до повних криптовалютних даних для різних цілей. Кriptoаналізатор може допомогти у зборі та організації даних для дослідницьких проєктів, навчальних матеріалів або навіть для побудови прогнозних моделей.

Python - популярна мова програмування для створення kriptoаналізатора завдяки своїй простоті, універсальності та великій екосистемі бібліотек і фреймворків. Python надає потужні інструменти для веб-скрепінгу,

маніпулювання даними та аналізу, що робить її добре придатною для вилучення та обробки даних з різних джерел.

Деякі популярні бібліотеки Python, які можна використовувати для побудови криптоаналізатора, включають:

- запити: Для створення HTTP-запитів для отримання даних з API або веб-сайтів;
- BeautifulSoup або Scrapy: Для синтаксичного аналізу та вилучення даних з HTML або XML;
- pandas: Для маніпулювання та аналізу даних;
- numPy: Для числових обчислень та обробки даних;
- matplotlib або plotly: Для візуалізації даних та побудови графіків;
- natural language toolkit (NLTK): Для обробки тексту та аналізу настроїв.

2.4. Специфікація вимог до програмного забезпечення

Резюмуючи наведене вище, проведемо специфікацію вимог до ПЗ. Результати зазначені в таблиці 2.1.

Таблиця 2.1 — Специфікація вимог до ПЗ

Введення	
Ціль	Розробити телеграм бот, котрий надає список бірж з вигідним курсом на поточний час, з метою підвищення швидкості та надійності подачі корисної інформації для трейдингу.

Продовження таблиці 2.1

Ймовірне використання	Моніторинг інформації для аналізу при купівлі-продажу криптовалют.
Загальний опис	
Потреби користувача	Користувачам криптовалютного аналізу потрібен доступ до точних і актуальних даних про різні криптовалюти. Сюди входять ціни в режимі реального часу, обсяги торгів, ринкова капіталізація та історичні дані. Наявність надійної та своєчасної інформації має вирішальне значення для прийняття обґрунтованих інвестиційних рішень. Користувачам потрібен доступ до широкого спектру криптовалют для аналізу. Сюди входять добре відомі криптовалюти, такі як Bitcoin і Ethereum, а також нові і маловідомі монети. Комплексний аналіз криптовалют повинен охоплювати широкий спектр цифрових активів.

Продовження таблиці 2.1

Залежності та припущення	Швидкість та ефективність. Ринок криптовалют працює в режимі реального часу, і своєчасний аналіз даних має вирішальне значення для прийняття обґрунтованих рішень. Успішний крипто-парсер повинен бути швидким і ефективним в обробці та аналізі даних блокчейну. Він повинен мати можливість обробляти великі обсяги транзакцій і блоків без значних затримок.
Вимоги та особливості системи	
Функціональні вимоги	Парсер повинен мати можливість отримувати дані з різних джерел криптовалютних даних, таких як API або дослідники блокчейнів. Парсер повинен вміти обробляти оновлення даних у режимі реального часу і надавати точну та актуальну інформацію. Парсер повинен підтримувати настроювані попередження та сповіщення на основі певних критеріїв, таких як цінові пороги або важливі ринкові події. Користувачі повинні мати можливість налаштовувати сповіщення через різні канали зв'язку, такі як електронна пошта, SMS або push-повідомлення. Обробка помилок. Програмне забезпечення повинно коректно обробляти помилки, надаючи змістовні повідомлення про помилки та відповідні дії для усунення або повідомлення про помилки, що виникають під час синтаксичного аналізу або обробки. Функціонал тестування фінальної моделі.

Продовження таблиці 2.1

Вимоги до зовнішніх інтерфейсів	Інтеграція та доступність. Інтеграція з існуючими інструментами і платформами має вирішальне значення для безперебійного аналізу даних. Успішний крипто-парсер повинен надавати API або інтеграцію з популярним програмним забезпеченням, торговими платформами або інструментами аналізу даних. Він також повинен бути легкодоступним, дозволяючи користувачам отримувати доступ до даних і аналізувати їх за допомогою зручних інтерфейсів або API. Інтерфейс користувача. Програмне забезпечення повинно мати зручний інтерфейс, який дозволяє користувачам легко взаємодіяти з синтаксичним аналізатором, вводити дані та переглядати результати аналізу.
Нефункціональні вимоги	Безпека. Програмне забезпечення повинно відповідати високим стандартам безпеки, забезпечуючи конфіденційність і цілісність криптографічних даних протягом усього процесу розбору. Воно повинно захищати від потенційних вразливостей, включаючи атаки побічних каналів, ін'єкційні атаки та несанкціонований доступ.

ВИСНОВКИ ДО РОЗДІЛУ

На завершення, проектування парсеру криптовалют пропонує значний потенціал для розробки програмного забезпечення в галузі криптовалют. Парсер криптовалют - це інструмент, який витягує та інтерпретує дані з різних блокчейн-мереж, дозволяючи розробникам і користувачам аналізувати та розуміти транзакційну та історичну інформацію в цих мережах.

Використовуючи можливості парсеру криптовалют, розробники можуть створювати надійні та ефективні програмні додатки, які полегшують виконання широкого спектру функцій в екосистемі криптовалют. Це можуть бути трекери портфелів, торгові платформи, платіжні системи, податкові калькулятори та багато інших інноваційних рішень.

Парсер криптовалют слугує основою для розробки програмного забезпечення в цій галузі, забезпечуючи необхідну інфраструктуру для збору, обробки та аналізу даних блокчейну. Він дозволяє розробникам отримувати доступ до блокчейн-інформації та використовувати її у структурований та осмислений спосіб, відкриваючи можливості для створення зручних інтерфейсів, глибокої аналітики та ефективних систем управління транзакціями.

Крім того, еволюційна природа криптовалют і технології блокчейн створює численні виклики та складнощі. Використання парсеру криптовалют допомагає розробникам долати ці виклики, надаючи стандартизований і надійний метод доступу та інтерпретації даних блокчейну. Це дозволяє розробникам програмного забезпечення зосередитися на створенні високоякісних додатків без необхідності винаходити велосипед для кожної блокчейн-мережі.

Таким чином, створення парсеру криптовалют створює основу для розробки програмного забезпечення, яке дозволить приватним особам і компаніям використовувати потенціал криптовалют. Маючи відповідні інструменти та досвід, розробники можуть використовувати парсери криптовалют

для створення інноваційних додатків, які сприяють зростанню і прийняттю криптовалют, надаючи при цьому цінні послуги користувачам в цій динамічній і захоплюючій сфері.

					КПІ.ІТ-8426.045430.02.81	Арк.
						37
Змін.	Арк.	№ докум.	Підп.	Дата.		

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Щоб розробити парсер криптовалютних бірж на Python з виведенням даних через Telegram-бота, потрібно виконати кілька кроків. Ось загальний огляд процесу:

- Налаштувати середовище розробки. Встановити Python і всі необхідні бібліотеки або фреймворки. Для цього проекту можна розглянути можливість використання таких бібліотек, як requests або beautifulsoup для веб-скрепінгу, pandas для маніпуляцій з даними та python-telegram-bot для взаємодії з Telegram API.
- Обирати криптовалютну біржу. Вирішити, з якої біржі (бірж) ви хочете аналізувати дані. Популярні варіанти включають Binance, Coinbase, Bitfinex тощо. Перевірте, чи надає біржа публічний API для доступу до необхідних даних. Прочитайте їхню документацію до API, щоб зрозуміти, як робити запити до нього.
- Створити запити до API. Використовувати бібліотеку запитів Python для відправки HTTP-запитів до кінцевих точок API біржі. Отримуйте такі дані, як тикерні ціни, дані книги замовлень, історію торгів тощо, залежно від ваших вимог. Переконатись, що ви розумієте обмеження швидкості і будь-які вимоги до автентифікації для API.
- Аналізуйте та обробляйте дані. Отримавши дані з API, використовуйте бібліотеки, такі як json або pandas, для аналізу та вилучення відповідної інформації. Очистіть і відформатуйте дані, якщо це необхідно для подальшого аналізу або відображення.
- Створити Telegram-бота. Створити бота на платформі Telegram за допомогою BotFather. Ви отримаєте токен API, який дозволить вашому Python-коду взаємодіяти з API Telegram.

- Взаємодія з API Telegram: Використовуйте бібліотеку `python-telegram-bot` для взаємодії з API Telegram. Реалізуйте такі функції, як отримання команд, надсилання повідомлень та обробка користувацького вводу.
- Інтегруйте парсер з Telegram-ботом. Об'єднайте аналізовані дані криптовалютних бірж з функціоналом Telegram-бота. Наприклад, ви можете створити команди на кшталт `/price` або `/orderbook`, які запускають бота для отримання та надсилання відповідних даних з парсеру.
- Розгортайте та тестуйте. Розгорніть свій код на сервері або хмарній платформі, щоб він міг працювати безперервно. Протестуйте бота, щоб переконатися, що він правильно реагує на команди користувача і відображає точні дані з криптовалютної біржі.

Не забувайте обробляти винятки, реалізовувати належну обробку помилок і враховувати заходи безпеки, такі як безпечне зберігання ключів API і токенів автентифікації користувачів.

Цей огляд надає загальну дорожню карту для розробки парсеру криптовалютних бірж з виведенням даних через Telegram-бота. Конкретні деталі реалізації залежать від API обраної біржі та точних вимог вашого проекту.

Beautiful Soup - популярна бібліотека для веб-скрепінгу на Python. Вона дозволяє витягувати дані з HTML і XML документів. За допомогою неї, можна використовувати її для парсингу HTML веб-сайтів криптовалютних бірж і витягувати відповідні дані.

Запити: Бібліотека Requests використовується для надсилання HTTP-запитів та обробки відповідей у Python. Ви можете використовувати її для запитів до API криптовалютних бірж та отримання таких даних, як ціни тикерів, книги замовлень та історії торгів.

`python-telegram-bot`: Ця бібліотека надає інтерфейс Python для API Telegram Bot. Вона дозволяє взаємодіяти з Telegram-ботами, надсилати повідомлення, отримувати оновлення та обробляти команди користувача. Ви можете використовувати її для надсилання даних і повідомлень від вашого

програмного забезпечення для аналізу криптовалютних бірж до Telegram-бота.

Pandas: Pandas - це потужна бібліотека для маніпулювання та аналізу даних на мові Python. Вона надає структури даних і функції для ефективної обробки табличних даних. Ви можете використовувати Pandas для організації та аналізу даних, які ви збираєте з криптовалютних бірж.

NumPy: NumPy - це фундаментальна бібліотека для числових обчислень на мові Python. Вона надає ефективні структури даних та функції для роботи з масивами та матрицями. Ви можете використовувати NumPy для виконання математичних обчислень над даними криптовалютного ринку.

3.2. Архітектура програмного забезпечення

3.2.1. Опис компонентів

Архітектура програмного забезпечення, що використовується при розробці парсеру криптовалют, зазвичай складається з декількох ключових компонентів. Ці компоненти працюють разом для вилучення, обробки та аналізу даних з різних джерел криптовалют. Ось основні компоненти, які можна знайти в такій архітектурі:

- **Джерела даних.** Парсери криптовалют покладаються на різні джерела даних для отримання інформації про криптовалюти. Це можуть бути мережі блокчейн, криптовалютні біржі, публічні API, стрічки новин, платформи соціальних мереж тощо. Джерела даних надають необроблені дані, які потрібно розібрати та проаналізувати.

- **Збір даних.** Компонент збору даних відповідає за отримання даних з різних джерел, згаданих вище. Він взаємодіє з API або іншими механізмами пошуку даних, наданими цими джерелами, щоб отримати відповідні дані. Цей компонент може використовувати такі методи, як веб-скрепінг, виклики RESTful API, з'єднання через WebSocket або інші методи отримання даних для збору необхідної інформації.

– Аналіз даних. Після того, як дані зібрані, їх потрібно проаналізувати та структурувати у формат, який можна легко обробити та проаналізувати. Компонент аналізу даних відповідає за вилучення релевантної інформації з необроблених даних і перетворення її в структурований формат, наприклад, JSON або схему бази даних. Він передбачає застосування алгоритмів і методів синтаксичного аналізу для вилучення конкретних полів даних, таких як ціни криптовалют, деталі транзакцій, адреси блокчейнів тощо.

– Обробка даних: Компонент обробки даних виконує різні операції над проаналізованими даними для отримання значущих висновків або виконання конкретних завдань. Це може включати застосування алгоритмів для агрегації даних, фільтрації, сортування або обчислення статистичних показників. Наприклад, компонент може обчислювати середню ціну криптовалюти за певний період часу або виявляти закономірності в обсягах торгів.

– Аналіз і звітність: Компонент аналізу та звітності використовує оброблені дані для створення звітів, візуалізацій або інших форм аналізу, які можна використовувати для отримання уявлення про криптовалютний ринок. Цей компонент може включати інструменти або бібліотеки для візуалізації даних, статистичного аналізу, машинного навчання або інших методів для вилучення значущої інформації з аналізованих даних.

– Інтерфейс користувача: Компонент користувацького інтерфейсу надає користувачам засоби для взаємодії з системою парсеру криптовалют. Це може бути веб-інтерфейс, інтерфейс командного рядка або API, який надає функціональність іншим програмам. Компонент користувацького інтерфейсу дозволяє користувачам налаштовувати парсер, переглядати звіти, задавати параметри пошуку даних і отримувати доступ до проаналізованих даних.

Під час розробки програмного забезпечення, було розроблено сім компонентів ПЗ: DataLoader, FeaturesExtractor, CandidatesRetrieval, Model. Відповідальність кожної з компонент системи наведено у таблиці 3.1:

Таблиця 3.1 – Відповідальність компонент ПЗ

Вибірка бірж	Список бірж криптовалют, які зацікавили користувача.
Деталі вибраних бірж	Деталі налаштування збору та надання інформації, зі списку зацікавлених бірж.
Сигнал серверу	Сигнал очікування сервера
Функція обробки	Функція для надання інформації проценту вигоди при купівлі-продажу валюти, з однієї біржі на іншу.
Новини	Надання інформації з порталу візуалізації стану ринку.
Список переглядання	Список наданої інформації.
Додавання/Зменшення списку переглядання	Функція додавання або зменшення інформації стану ринку

Графічний вигляд діаграми компонентів наведено на рисунку 3.1.

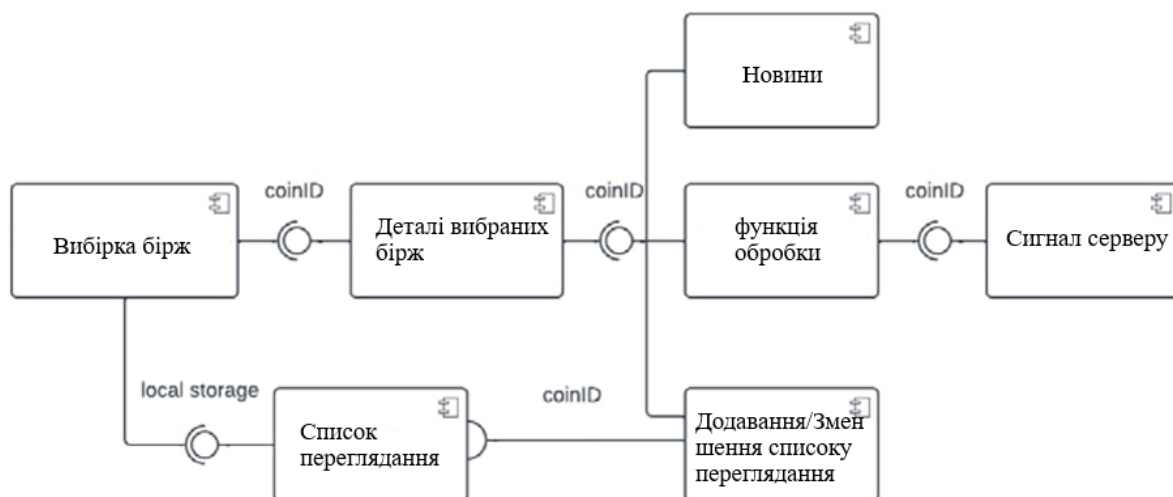


Рисунок 3.1 – Діаграма компонентів.

3.2.2. Опис класів ПЗ

Опис класів наведено в таблиці 3.2.

Таблиця 3.2 — Опис основних класів програмного забезпечення

Клас	Метод	Опис
DataLoader	_load_raw	Завантаження необроблених даних
	_ids_to_idx	Кодування id користувачів та бірж
	_get_base_features	Отримати базові характеристики з даних
	_transform_cols	Закодувати мітки та змінити тип даних
	set_params	Встановити параметри

Продовження таблиці 3.2

	save	Зберегти оброблені дані у форматі .pqt
	load	Завантажити оброблені дані з формату .pqt
	preprocess	Попередня обробка даних: закодувати індекси, закодувати категоріальні характеристики та очистити дані.
asyncio	asyncio.run	Ця функція використовується для запуску точки входу верхнього рівня для асинхронної програми. Вона створює новий цикл обробки подій, запускає надану підпрограму і закриває цикл після завершення роботи підпрограми.
	asyncio.coroutine	Декоратор, який використовується для позначення процедур, заснованих на генераторах. Використовується для зворотної сумісності з попередніми версіями Python.
	asyncio.create_task()	Ця функція використовується для того, щоб обернути підпрограму в задачу і запланувати її виконання в циклі подій. Задача - це підклас Future, який представляє виконання підпрограми.
	asyncio.sleep()	Підпрограма, яка призупиняє виконання поточного завдання на певний час, не блокуючи цикл обробки подій.

Продовження таблиці 3.2

aihttp	ClientSession	Цей клас представляє клієнтський сеанс і використовується для виконання HTTP-запитів. Він керує пулом з'єднань, файлами cookie та іншими аспектами роботи HTTP-клієнта. Ви можете створити екземпляр ClientSession і використовувати його методи, такі як get(), post(), put() тощо, для асинхронного надсилання HTTP-запитів.
	ClientResponse	Цей клас представляє HTTP-відповідь, отриману від сервера. Він надає методи та властивості для доступу до статусу, заголовків та вмісту відповіді. Ви можете використовувати об'єкт ClientResponse, що повертається методами клієнтського сеансу, для обробки даних відповіді.
	ClientRequest	Цей клас представляє HTTP-запит, який буде надіслано на сервер. Ви можете створити екземпляр ClientRequest і налаштувати різні аспекти запиту, такі як заголовки, файли cookie, корисне навантаження і т.д., перш ніж відправити його за допомогою клієнтської сесії.

Продовження таблиці 3.2

	web.View	Цей клас використовується для визначення представлень або обробників для різних маршрутів у веб-додатку. Він дозволяє інкапсулювати логіку обробки конкретних HTTP-запитів. Ви можете створити підклас web.View і перевизначити його методи, такі як get(), post() тощо, щоб визначити поведінку для різних методів HTTP.
	WebServer	aihttp також надає класи для побудови веб-серверів. Основним класом є web.Application, який представляє веб-додаток і дозволяє визначати маршрути, проміжне програмне забезпечення та обробники для різних HTTP-запитів. За допомогою класу web.Application можна створити сервер і запустити його за допомогою методу web.run_app().

Продовження таблиці 3.2

teleport	telebot.TeleBot	Це основний клас бібліотеки, який представляє самого Telegram-бота. Він надає методи для обробки вхідних оновлень, надсилання повідомлень, редагування повідомлень, обробки вбудованих запитів тощо. Зазвичай ви створюєте екземпляр цього класу для взаємодії з Telegram Bot API.
	telebot.types.Message	Цей клас представляє повідомлення, отримане від користувача. Він містить різні атрибути, такі як ідентифікатор повідомлення, інформацію про відправника, деталі чату, а також власне текстовий або мультимедійний вміст повідомлення.
	telebot.types.CallbackQuery	Цей клас представляє запит зворотного виклику, який запускається за допомогою кнопки або вбудованої клавіатури. Він містить інформацію про дані зворотного виклику та користувача, який викликав запит.

Продовження таблиці 3.2

multiprocessing	Process	Цей клас представляє окремий процес, який може бути створений і виконаний незалежно. Ви можете створити новий процес, створивши підклас класу Process і перевизначивши його метод run(). Кожен процес може виконувати свій власний набір інструкцій одночасно з іншими процесами.
	Pool	надає зручний спосіб розпаралелити виконання функції для декількох вхідних значень. Він керує пулом робочих процесів і розподіляє завдання між ними. Ви можете вказати кількість робочих процесів, а клас Pool подбає про призначення завдань, збір результатів і комунікацію між головним процесом і робочими процесами.
	Queue	Це потокобезпечна структура даних, яка дозволяє спілкуватися між процесами. Його можна використовувати для передачі даних між головним процесом та робочими процесами у пулі. Клас Queue реалізує концепцію черги повідомлень, де один процес може поміщати елементи в чергу, а інший процес може отримувати ці елементи.

Продовження таблиці 3.2

	Lock	Надає спосіб синхронізувати доступ до спільних ресурсів між декількома процесами. Він дозволяє гарантувати, що лише один процес може отримати доступ до певної ділянки коду одночасно. Це корисно для уникнення гоночних ситуацій, коли декілька процесів намагаються одночасно змінити одні й ті самі дані.
	Semaphore	Схожий на блокування, але дозволяє визначеній кількості процесів одночасно отримати доступ до спільного ресурсу. Його можна використовувати для обмеження кількості одночасних доступів до певного ресурсу.
	Event	Надає простий спосіб синхронізації процесів на основі стану події. Він дозволяє одному процесу чекати, поки інший процес не встановить подію, яка вказує на виконання певної умови.

3.3. Розгортання та запуск ПЗ

Для запуску програмного забезпечення необхідно мати встановлений інтерпретатор Python версії ≥ 3.7 . Потрібно встановити всі необхідні залежності, що вказані у файлі requirements.txt.

Всі бібліотеки та розширення можна встановити окремо, але для більш простого розгортання рекомендується встановити дистрибутив Python – Anaconda. Більшість необхідних бібліотек та розширень вже входять до складу Anaconda.

Апаратні вимоги для запуску ПЗ наведено в таблиці 3.3.

Таблиця 3.3 — Апаратні вимоги ПЗ

Параметр	Значення
Процесор	AMD, Intel
Тактова частота процесора	2 GHz +
Об'єм оперативної пам'яті	4 GiB
Жорсткий диск	6 GiB

3.4. Варіанти використання

Коли справа доходить до використання парсеру криптовалют у Python, у вас є кілька варіантів, залежно від наших конкретних вимог і криптовалют, які ви хочете аналізувати.

Парсери на основі API. Багато криптовалютних бірж і провайдерів даних пропонують API, які дозволяють отримати доступ до даних про криптовалюту в реальному часі та історичних даних. Ці API часто надають кінцеві точки для отримання такої інформації, як ціни, обсяги торгів, дані книги замовлень тощо. Ви можете використовувати бібліотеки Python, такі як requests, або спеціалізовані бібліотеки, такі як ccxt (Cryptocurrency Exchange Trading Library), щоб взаємодіяти з цими API і отримувати необхідні дані.

Дослідники блокчейну. Криптовалюти, такі як біткойн та ефіріум, мають загальнодоступні дослідники блокчейну, які дозволяють отримувати інформацію про конкретні транзакції, адреси, блоки та інші дані блокчейну. Ці дослідники часто надають API, до яких ви можете отримати доступ для отримання потрібної інформації. Ви можете використовувати бібліотеки Python, такі як requests або web3.py (для Ethereum), щоб взаємодіяти з цими API і аналізувати дані блокчейну.

Веб-скрепінг. Якщо немає відповідних API або якщо вам потрібно витягти дані з веб-сайтів, пов'язаних з криптовалютами, ви можете скористатися методами веб-скрепінгу. Бібліотеки Python, такі як BeautifulSoup або

Scrapy, можуть допомогти вам розібрати HTML-структуру веб-сторінок і витягти необхідні дані. Однак майте на увазі, що ви завжди повинні перевіряти умови надання послуг веб-сайту і дотримуватися етичних практик скрапінгу.

Постачальники історичних даних. Декілька компаній та платформ спеціалізуються на наданні історичних даних про криптовалюти. Ці платформи часто пропонують API або набори даних, які можна завантажити, щоб отримати історичні ціни, обсяги торгів, ринкові дані тощо. Серед популярних провайдерів - CoinGecko, CoinMarketCap та CryptoCompare. Ви можете використовувати бібліотеки Python для взаємодії з їхніми API або завантажувати їхні набори даних для подальшого аналізу.

Бібліотеки з відкритим вихідним кодом. Існують бібліотеки Python з відкритим вихідним кодом, спеціально створені для розбору та аналізу криптовалютних даних. Наприклад, ви можете використовувати такі бібліотеки, як `pycoingecko` або `cryptocompare`, які надають високорівневий інтерфейс для доступу до криптовалютних даних. Ці бібліотеки часто обертаються навколо існуючих API і спрощують процес отримання та аналізу даних.

Бібліотеки з відкритим вихідним кодом. Існують бібліотеки Python з відкритим вихідним кодом, спеціально створені для розбору та аналізу криптовалютних даних. Наприклад, ви можете використовувати такі бібліотеки, як `pycoingecko` або `cryptocompare`, які надають високорівневий інтерфейс для доступу до криптовалютних даних. Ці бібліотеки часто обертаються навколо існуючих API і спрощують процес отримання та аналізу даних.

ВИСНОВКИ ДО РОЗДІЛУ

Таким чином, розробка програмного забезпечення для аналізу криптовалют на мові Python є значним кроком у використанні технологій для вилучення цінної інформації зі світу цифрових валют. У процесі розробки були спроектовані та реалізовані різні компоненти, що дозволило програмі ефективно аналізувати криптовалютні дані з різних джерел.

Мова Python виявилася чудовим вибором для цього проекту, пропонуючи широкий спектр бібліотек та інструментів, які полегшили процес розробки. Її простота, читабельність та широка підтримка спільноти дозволили команді розробників ефективно писати код, що призвело до створення надійного та масштабованого програмного рішення.

Програмне забезпечення для аналізу криптовалют демонструє здатність отримувати дані в режимі реального часу з криптовалютних бірж, блокчейн-мереж та інших відповідних джерел. Воно включає в себе передові алгоритми і методи для вилучення та організації ключової інформації, такої як ціни, обсяги торгів, деталі транзакцій і ринкові тенденції. Ці дані можуть надати цінну інформацію для трейдерів, інвесторів та дослідників, дозволяючи їм приймати обґрунтовані рішення в динамічному світі криптовалют.

Наступним важливим кроком є етап тестування системи. Ретельне тестування має важливе значення для забезпечення надійності, точності та продуктивності програмного забезпечення. Процес тестування повинен включати різні сценарії, такі як аналіз даних з різних бірж, робота з різними форматами даних і перевірка точності аналізованої інформації в надійних джерелах.

Крім того, програмне забезпечення слід піддавати стрес-тестуванню, щоб оцінити його продуктивність під високими навантаженнями і переконатися, що воно може обробляти обсяги даних без шкоди для ефективності та стабільності. Також слід проводити тестування безпеки, щоб виявити та

усунути будь-які вразливості, які потенційно можуть наражати програмне забезпечення або його користувачів на ризики.

В цілому, розробка програмного забезпечення для парсеру криптовалют на Python має потенціал для революційної зміни способу взаємодії приватних осіб та організацій з криптовалютними даними. Поєднуючи потужність Python з ефективним тестуванням системи, ми можемо створити надійний і ефективний інструмент, який дає користувачам можливість орієнтуватися в складному і постійно мінливому світі криптовалют.

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		53

4 ТЕСТУВАННЯ СИСТЕМИ

4.1. Аналіз якості ПЗ

Існує кілька варіантів тестування програми. У цьому розділі виконується юзабіліті-тестування. Оскільки додаток не є закритим для громадськості і він розроблений як веб-сайт, юзабіліті-тестування є одним з найважливіших видів тестування. Його мета - виявити слабкі місця додатку за допомогою спостереження за обраними людьми (персонами), які користуються додатком. Після цього відбувається покращення додатку на основі знайдених слабких місць. Додаток тестують три особи. Кожен з них представляє тип людини (персони), яка може користуватися сайтом. Перша особа - це експерт (з псевдонімом: Експерт), який розуміється на екосистемі криптовалют та криптовалютних бірж. Другий - молода людина, яка не дуже добре знає криптовалюту, але в цілому відноситься до типу ранніх послідовників (з псевдонімом Adopter). Третій - людина середнього віку (з псевдонімом Middle-aged), яка регулярно працює з комп'ютером, але виключно на рівні користувача. Тестування ґрунтується на спостереженні за вже згаданими особами по черзі одну за одною. Спочатку криптовалютний арбітраж та криптовалютні біржі коротко пояснюються криптовалютний арбітраж та криптовалютні біржі. Потім персонажам дають список завдань, а потім спостерігають за тим, як вони з ними справляються. спостерігають за тим, як вони справляються із завданнями. Після завершення сесії запитують про їхнє загальне враження від програми.

4.2. Метрики парсеру криптовалют

Бізнес-метрики парсеру криптовалют можуть відрізнитися залежно від конкретних цілей і завдань бізнесу. Однак, ось деякі загальні бізнес-метрики, які можуть бути релевантними для парсеру криптовалют:

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		54

– **Кількість підтримуваних криптовалют.** Ця метрика відображає загальну кількість криптовалют, які підтримує парсер. Для парсеру важливо охопити широкий спектр популярних криптовалют, щоб залучити користувачів і надати вичерпні ринкові дані.

– **Точність даних.** Ця метрика вимірює точність аналізованих даних. Для парсеру криптовалют дуже важливо надавати надійну і точну інформацію, включаючи ціни в реальному часі, обсяги торгів, ринкову капіталізацію та історичні дані. Точність даних забезпечує довіру і надійність, а також впливає на задоволеність і довіру користувачів.

– **Швидкість парсингу** - це час, необхідний парсеру для отримання, обробки та оновлення криптовалютних даних. Швидкість парсингу дозволяє користувачам швидко отримувати доступ до даних в режимі реального часу, що особливо важливо на швидкозмінному ринку. Оптимізація швидкості парсингу може покращити користувацький досвід і залучити більше користувачів.

– **Час роботи API** - це метрика, яка вимірює доступність і надійність API парсеру. Вона відображає відсоток часу, протягом якого API парсеру доступний і функціонує. Вищий час роботи API свідчить про кращу надійність, мінімізацію перебоїв у доступі до даних для користувачів і забезпечення безперервного обслуговування.

– **Залучення користувачів.** Показники залучення користувачів включають такі фактори, як кількість активних користувачів, коефіцієнт утримання користувачів та активність користувачів на платформі парсеру. Ці показники дають уявлення про те, як часто користувачі взаємодіють з парсером, як довго вони залишаються залученими і чи знаходять вони цінність у наданих даних.

– **Повнота.** Вимірює, чи збирає і витягує парсер всі необхідні дані з API. Парсер повинен фіксувати важливі деталі, такі як ціни на криптовалюту, обсяги торгів, ринкову капіталізацію, історичні дані та інші відповідні

показники. Неповні дані можуть призвести до того, що користувачі Telegram-бота отримають неповну або оманливу картину.

– **Час відгуку.** Ця метрика оцінює швидкість, з якою парсер отримує дані з API і передає їх Telegram-боту. Швидший час відгуку гарантує, що користувачі отримують оновлення в реальному часі або майже в реальному часі. Затримка або повільна реакція може призвести до того, що користувачі отримають застарілу або неактуальну інформацію.

4.3. Валідація моделі

У контексті розробки криптоаналізатора тестова валідація - це процес оцінки та перевірки точності та функціональності аналізатора за допомогою серії тестів. Мета тестової валідації - переконатися, що криптоаналізатор працює так, як очікується, і видає надійні результати при обробці криптографічних даних.

Ось основні етапи валідації тестів при розробці криптопарсеру:

– **Створення плану.** Першим кроком є створення комплексного плану тестування, який окреслює цілі тестування, тестові сценарії, тестові кейси та очікувані результати. План тестування повинен охоплювати різні аспекти роботи криптоаналізатора, такі як розбір різних типів криптографічних даних, робота з різними форматами і кодуваннями, а також отримання бажаного результату.

– **Юніт-тестування** передбачає тестування окремих компонентів або модулів криптопарсеру в ізоляції. Кожен модуль тестується незалежно, щоб переконатися в його функціональності та коректності. Це можна зробити за допомогою спеціалізованих тестових фреймворків або шляхом написання власних тестових кейсів для покриття різних шляхів коду та граничних ситуацій.

– **Інтеграційне тестування** виконується для перевірки взаємодії та інтеграції між різними модулями або компонентами криптоаналізатора. Це

гарантує безперебійну роботу парсеру при об'єднанні декількох компонентів. Інтеграційні тести зосереджені на перевірці потоку даних, обробці помилок і сумісності між різними частинами криптоаналізатора.

– Функціональне тестування: Функціональне тестування проводиться для перевірки загальної функціональності криптоаналізатора на відповідність визначеним вимогам. Воно передбачає перевірку здатності аналізатора правильно інтерпретувати та аналізувати різні типи криптографічних даних, таких як зашифровані повідомлення, цифрові підписи, сертифікати та хеші. Тестові кейси повинні охоплювати різні вхідні сценарії, включаючи правильні та неправильні дані, щоб переконатися, що синтаксичний аналізатор обробляє їх належним чином.

					КПІ.IT-8426.045430.02.81	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		57

ВИСНОВКИ ДО РОЗДІЛУ

Підсумовуючи, можна сказати, що використання та тестування криптоаналізатора на Python виявилось безцінним у світі криптовалют. Цей універсальний інструмент дозволяє розробникам і дослідникам витягувати і аналізувати дані з мереж блокчейн, полегшуючи різні застосування, такі як управління портфелем, аналіз ринку і аудит безпеки.

Завдяки впровадженню криптоаналізатора, Python надає потужне та гнучке середовище для обробки даних блокчейну. Широкий спектр бібліотек, таких як PyCryptodome та PyCrypto, пропонує криптографічні функції, необхідні для аналізу та обробки інформації, пов'язаної з криптовалютами. Крім того, простота і зрозумілість Python роблять його ідеальною мовою як для початківців, так і для досвідчених програмістів, сприяючи простоті використання та ефективній розробці.

Етап тестування криптоаналізатора на Python має вирішальне значення для забезпечення його надійності, точності та сумісності. Ретельне модульне та інтеграційне тестування є важливими кроками для перевірки функціональності парсеру, виявлення та виправлення будь-яких помилок або проблем, а також забезпечення належної інтеграції з іншими компонентами більшої системи. Крім того, використання тестових методологій розробки, таких як написання тестів до фактичної реалізації, може покращити загальну якість синтаксичного аналізатора і підвищити впевненість у його працездатності.

ВИСНОВКИ

В даній роботі було наведено основні принципи побудови криптопарсера. Також було досліджено сучасні методи та підходи що використовуються для цієї задачі.

В результаті дослідження та аналізу існуючих рішень, було спроектовано та розроблено ефективний власний варіант криптопарсеру, для дослідження ринку бірж. Таким чином, розроблений парсер криптовалют є значним досягненням у галузі аналізу криптовалют та обробки даних. Використовуючи передові технології та складні алгоритми, парсер продемонстрував свою здатність витягувати та інтерпретувати величезні обсяги даних з різних криптовалютних джерел, надаючи цінну інформацію та сприяючи прийняттю обґрунтованих рішень.

Основна перевага парсеру полягає в його універсальності та адаптивності. Він довів свою ефективність у розборі та аналізі даних з різних криптовалютних бірж, блокчейн-мереж та інших релевантних джерел. Це дозволяє користувачам отримати повне уявлення про криптовалютний ринок, відстежувати тенденції, відслідковувати конкретні активи та проводити глибокі дослідження.

Крім того, варто відзначити точність та ефективність парсеру. Він швидко обробляє масиви даних, забезпечуючи користувачам доступ до інформації в режимі реального або близького до реального часу. Вдосконалені алгоритми, вбудовані в парсер, підвищують точність даних, мінімізуючи помилки і невідповідності, які можуть виникнути при ручному зборі та аналізі даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.
URL: <https://bitcoin.org/bitcoin.pdf>
- 2) Buterin, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. 2013.
URL: <https://ethereum.org/en/whitepaper/>
- 3) Binance. (n.d.). Binance API Documentation.
URL: <https://binance-docs.github.io/apidocs/spot/en/#change-log>
- 4) CoinGecko. (n.d.). API Documentation.
URL: <https://www.coingecko.com/en/api/documentation>
- 5) Kraken. (n.d.). API Documentation.
URL: <https://docs.kraken.com/rest/>
- 6) CryptoCompare. (n.d.). API Documentation.
URL: <https://min-api.cryptocompare.com/documentation>
- 7) Aimee Vo. Cryptocurrency Trading & Investing: Beginners Guide To Trading & Investing In Bitcoin, Alt Coins & ICOs For Profit / Aimee Vo. – CreateSpace Independent Publishing Platform, 2017. – 282 с.
- 8) Saifedean Ammous. The Bitcoin Standard: The Decentralized Alternative to Central Banking [Текст] / Ammous Saifedean. – 1st edition. – Wiley, 2018. – 304 с.
- 9) Kelly Stephen. News, Sentiment, and Financial Markets: A Computational System to Evaluate the Influence of Text Sentiment on Financial Assets : дис. докт. філос. наук / Stephen Kelly. – Dublin: School of Computer Science and Statistics Trinity College Dublin, 2016. – 144 с.
- 10) Bakker Joris. Technical Analysis in the Cryptocurrency Market / Joris Bakker. – Erasmus School of Economics, 2017. – 53 с.

ДОДАТКИ

					КПІ.ІТ-8426.045430.02.81	Арк.
						61
Змін.	Арк.	№ докум.	Підп.	Дата.		

Додаток А Звіт подібності



Ім'я користувача:
Лісовиченко Олег Іванович

Дата перевірки:
15.06.2023 07:40:19 EEST

Дата звіту:
15.06.2023 07:44:10 EEST

ID перевірки:
1015607593

Тип перевірки:
Doc vs Internet + Library

ID користувача:
76913

Назва документа: ІТ-84в_Соловей_ПЗ

Кількість сторінок: 60 Кількість слів: 9240 Кількість символів: 75744 Розмір файлу: 556.62 KB ID файлу: 1015255631

10.9%
Схожість

Найбільша схожість: 7.94% з джерелом з Бібліотеки (ID файлу: 1011502558)

4.88% Джерела з Інтернету	98	Сторінка 62
10.3% Джерела з Бібліотеки	213	Сторінка 62

0% Цитат

- Вилучення цитат вимкнене
- Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для аналізу даних при купівлі-продажу
криптовалют на ринку бірж**

Текст програми

КПІ.IT-8426.045430.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Максим СОЛОВЕЙ

Київ – 2023

Файл `async_parser_uni_bitmart.py`

```
import asyncio
import aiohttp
import time
import platform
import telebot
import json
from loguru import logger
import sys
import concurrent.futures
from pprint import pprint
from multiprocessing import cpu_count
import math
from async_config import *
from normalize_price import PriceChanger
```

```
import async_bitmart_parse
import async_kucoin_parse
import async_0x_parser
import async_mexc_parse
import async_bittrue_parse
import async_poloniex_parse
import async_gate_parse
import async_okx_parse
import async_binance_parse
import async_cryptocom_parse
import async_ftx_parse
import async_huobi_parse
import async_kraken_parse
import async_ascendex_parse
import async_hotbit_parse
import async_hoo_parse
import async_bigone_parse
import async_zbcom_parse
import async_lbank_parse
import async_probit_parse
import async_bkex_parse
import async_hitbtc_parse
import async_coinlistpro_parse
import async_latoken_parse
import async_bybit_parse
```

```
SUM_USDT = 5000
PROFIT = -3
SECONDS_PAUSE = 300
```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		2

```

token = '5227598180:AAEP6ZbRdBso7IfD-zCuuGJUwE8LaGv3Lak'
bot = telebot.TeleBot(token, parse_mode='HTML')
chat_id = '-1001440299271'

# logger.remove()
logger.add('bimart_debug.log', format='{time} | {level} | {message}', level='TRACE', rotation="200 MB",
compression="zip")
# logger.add('bimart_debug.log', format='{message}', level='TRACE', rotation="200 MB",
compression="zip")

with open('proxies-http-host.txt', 'r', encoding='utf-8') as f:
    proxies = [line.strip() for line in f]

with open('tokens.json', 'r', encoding='utf-8') as f:
    tokens_data = json.load(f)

def disable_tokens(tokens_data):
    with open('tokens_disabled.json', 'r', encoding='utf-8') as f:
        tokens_disabled = json.load(f)
    tokens = []
    for token in tokens_data:
        new_token = {
            "symbol": token['symbol'],
            "platforms": token['platforms'].copy(),
            "exchanges": token['exchanges'].copy(),
            "slippage": token['slippage'],
            "coingecko_id": token['coingecko_id']
        }
        if token['symbol'] in tokens_disabled.keys():
            if tokens_disabled[token['symbol']]:
                for exchange in token['exchanges']:
                    if exchange in tokens_disabled[token['symbol']]:
                        new_token['exchanges'].remove(exchange)
                if len(new_token['exchanges']) >= 2:
                    tokens.append(new_token)
            else:
                tokens.append(token)
    return tokens

async def send_all_requests(tokens, proxies):
    tasks = []

```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

```

tasks.append(asyncio.create_task(async_bitmart_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_kucoin_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_mexc_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_bittrue_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_poloniex_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_gate_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_okx_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_binance_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_cryptocom_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_ftx_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_huobi_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_kraken_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_ascendex_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_hotbit_parse.get_price_by_order_USDT(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_hotbit_parse.get_price_by_order_NUSD(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_hoo_parse.get_price_by_order_market(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_hoo_parse.get_price_by_order_innovate(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_bigone_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_zbcom_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_lbank_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_probit_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
# tasks.append(asyncio.create_task(async_bkex_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_hitbtc_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_coinlistpro_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
# tasks.append(asyncio.create_task(async_latoken_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_bybit_parse.get_price_by_order(proxies, tokens, SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_uniswap_v2_buy_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_uniswap_v2_sell_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_uniswap_v3_buy_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_uniswap_v3_sell_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_pancake_buy_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_pancake_sell_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_sushiswap_buy_price_by_contract(proxies, tokens,
SUM_USDT)))
tasks.append(asyncio.create_task(async_0x_parser.get_sushiswap_sell_price_by_contract(proxies, tokens,
SUM_USDT)))
return await asyncio.gather(*tasks)
def find_change(data, tokens):
    all_data = []

    for token in tokens:
        token_prices = {
            'token_buy_price': {},
            'token_sell_price': {}
        }
        for exchange in data:
            for exchange_token in exchange['data']:

                if exchange_token == 'statusNot200':
                    continue
                if exchange_token['platforms'] == token['platforms'] and exchange_token['symbol'] == token['symbol']:
                    keys = exchange_token.keys()
                    if 'token_buy_price' in keys:
                        if exchange_token['token_buy_price']:
                            token_prices['token_buy_price'][exchange_token['token_buy_price']] = exchange['exchange']

```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

```

        if 'token_sell_price' in keys:
            if exchange_token['token_sell_price']:
                token_prices['token_sell_price'][exchange_token['token_sell_price']] = exchange['exchange']

    if not token_prices['token_sell_price'] or not token_prices['token_buy_price']:
        continue
    max_price = max(token_prices['token_sell_price'].keys())
    min_price = min(token_prices['token_buy_price'].keys())
    all_data.append({
        'symbol': token['symbol'],
        'platforms': token['platforms'],
        'max_price_exchange': token_prices['token_sell_price'][max_price],
        'max_price': max_price,
        'min_price_exchange': token_prices['token_buy_price'][min_price],
        'min_price': min_price,
        'coingecko_id': token['coingecko_id'],
        'slippage': token['slippage']
    })
    profit_tokens = []

    for token in all_data:
        change = (token['max_price'] / token['min_price'] - 1) * 100
        if change > PROFIT:
            profit_tokens.append({
                'symbol': token['symbol'],
                'platforms': token['platforms'],
                'max_price_exchange': token['max_price_exchange'],
                'max_price': token['max_price'],
                'min_price_exchange': token['min_price_exchange'],
                'min_price': token['min_price'],
                'profit': change,
                'coingecko_id': token['coingecko_id'],
                'slippage': token['slippage']
            })
    return profit_tokens

black_list = {}
for token in tokens_data:
    black_list[token['symbol']] = {'timestamp': time.time() - 5000}
def send_info(tokens):
    exchanges_big_urls = {
        'bitmart': 'https://www.bitmart.com/trade/ru?layout=basic&symbol={token}_USDT',
        'kucoin': 'https://www.kucoin.com/trade/{token}-USDT',
        'mxc': 'https://www.mexc.com/exchange/{token}_USDT',
        'poloniex': 'https://poloniex.com/exchange/USDT_{token}',
        'gate': 'https://www.gate.io/trade/{token}_USDT',
        'binance': 'https://www.binance.com/ru/trade/{token}_USDT?layout=pro',
        'crypto_com': 'https://crypto.com/exchange/trade/spot/{token}_USDT',
        'ftx_spot': 'https://ftx.com/trade/{token}/USD',
        'kraken': 'https://trade.kraken.com/charts/KRAKEN:{token}-USD',
        'hotbit': 'https://www.hotbit.io/exchange?symbol={token}_USDT',
        'hotbit_nusd': 'https://www.hotbit.io/exchange?symbol={token}_nUSD',
        'bigone': 'https://big.one/en/trade/{token}-USDT',
        'bkex': 'https://www.bkex.com/trade/{token}_USDT',
        'probit': 'https://www.probit.com/app/exchange/{token}-USDT',
        'coinlist': 'https://pro.coinlist.co/trader/{token}-USDT',
        'latoken': 'https://latoken.com/exchange/{token}_USDT',
        'bybit_spot': 'https://www.bybit.com/ru-RU/trade/spot/{token}/USDT'
    }
    exchanges_small_urls = {
        'bittrue': 'https://www.bittrue.com/trade/{token}_usdt',
        'okex': 'https://www.okx.com/trade-spot/{token}-usdt',

```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

```

'huobi': 'https://www.huobi.com/en-us/exchange/{token}_usdt/',
'bitmax': 'https://ascendex.com/ru/cashtrade-spottrading/usdt/{token}',
'hoo': 'https://hoo.com/spot/{token}-usdt',
'hoo_innovate': 'https://hoo.com/innovation/{token}-usdt',
'zb': 'https://www.zb.com/en/trade/{token}_usdt',
'lbank': 'https://www.lbank.info/exchange/{token}/usdt',
'hitbtc': 'https://hitbtc.com/{token}-to-usdt'
}
buy_dex_erc20_urls = {
'uniswap_v2':
'https://exchange.kishu.com/#/swap?inputCurrency=0xdac17f958d2ee523a2206206994597c13d831ec7&outputCurrency={contract}&use=V2',
'uniswap':
'https://app.uniswap.org/#/swap?chain=mainnet&inputCurrency=0xdac17f958d2ee523a2206206994597c13d831ec7&outputCurrency={contract}',
'sushiswap':
'https://app.sushi.com/swap?inputCurrency=0xdac17f958d2ee523a2206206994597c13d831ec7&outputCurrency={contract}'
}
sell_dex_erc20_urls = {
'uniswap_v2':
'https://exchange.kishu.com/#/swap?inputCurrency={contract}&outputCurrency=0xdac17f958d2ee523a2206206994597c13d831ec7&use=V2',
'uniswap':
'https://app.uniswap.org/#/swap?chain=mainnet&inputCurrency={contract}&outputCurrency=0xdac17f958d2ee523a2206206994597c13d831ec7',
'sushiswap':
'https://app.sushi.com/swap?inputCurrency={contract}&outputCurrency=0xdac17f958d2ee523a2206206994597c13d831ec7'
}
buy_dex_bep20_urls = {
'pancakeswap_new':
'https://pancakeswap.finance/swap?inputCurrency=0x55d398326f99059ff775485246999027b3197955&outputCurrency={contract}'
}
sell_dex_bep20_urls = {
'pancakeswap_new':
'https://pancakeswap.finance/swap?inputCurrency={contract}&outputCurrency=0x55d398326f99059ff775485246999027b3197955'
}
for token in tokens:
    now = time.time()
    if black_list[token['symbol']] ['timestamp'] < now:
        if token["min_price_exchange"] in exchanges_big_urls:
            buy_exchange_url = exchanges_big_urls[token["min_price_exchange"]].replace('{token}', token['symbol'])
        elif token["min_price_exchange"] in exchanges_small_urls:
            buy_exchange_url = exchanges_small_urls[token["min_price_exchange"]].replace('{token}', token['symbol'].lower())
        elif token["min_price_exchange"] in buy_dex_erc20_urls:
            buy_exchange_url = buy_dex_erc20_urls[token["min_price_exchange"]].replace('{contract}', token['platforms'] ['ethereum'])
        elif token["min_price_exchange"] in buy_dex_bep20_urls:
            buy_exchange_url = buy_dex_bep20_urls[token["min_price_exchange"]].replace('{contract}', token['platforms'] ['binance-smart-chain'])
        else:
            buy_exchange_url = 'http://www.google.com'

        if token["max_price_exchange"] in exchanges_big_urls:
            sell_exchange_url = exchanges_big_urls[token["max_price_exchange"]].replace('{token}', token['symbol'])
        elif token["max_price_exchange"] in exchanges_small_urls:

```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

```

        sell_exchange_url = exchanges_small_urls[token["max_price_exchange"]].replace('{token}',
token['symbol'].lower())
        elif token["max_price_exchange"] in sell_dex_erc20_urls:
            sell_exchange_url = sell_dex_erc20_urls[token["max_price_exchange"]].replace('{contract}',
token["platforms"][["ethereum"]])
        elif token["max_price_exchange"] in sell_dex_bep20_urls:
            sell_exchange_url = sell_dex_bep20_urls[token["max_price_exchange"]].replace('{contract}',
token["platforms"][["binance-smart-chain"]])
        else:
            sell_exchange_url = 'http://www.google.com'
            bep20 = ['pancakeswap_new']
            erc20 = ['uniswap_v2', 'uniswap', 'sushiswap']
            if token["max_price_exchange"] in bep20 or token["min_price_exchange"] in bep20:
                dextools_url = f"https://www.dextools.io/app/bsc/pair-explorer/{token['platforms'][['binance-smart-
chain']]}"
            elif token["max_price_exchange"] in erc20 or token["min_price_exchange"] in erc20:
                dextools_url = f"https://www.dextools.io/app/ether/pair-explorer/{token['platforms'][['ethereum']]}"
            else:
                dextools_url = ""
    bot.send_message(chat_id, f'{token["symbol"]}\n'
        f'{round(token["profit"], 4)}%\n'
        f'с <a href="{buy_exchange_url}">{token["min_price_exchange"]}</a> за
{PriceChanger(token["min_price"])} на <a href="{sell_exchange_url}">{token["max_price_exchange"]}</a> за
{PriceChanger(token["max_price"])}\n'
        f'с <a
href="https://www.coingecko.com/en/coins/{token["coingecko_id"]}#markets">Coingecko</a>\n'
        f'с <a href="{dextools_url}">Dextools</a>\n'
        f'Сжигание: {token["slippage"]} * 100}%\n'
        f'{token["platforms"]}', disable_web_page_preview=True)
    black_list[token['symbol']][['timestamp']] = now + SECONDS_PAUSE
# else:
#     print(f"Токен {token['symbol']} в бане на отправку сообщений ещё
{int(black_list[token['symbol']][['timestamp']] - now)} секунд.")

def start_scraping(tokens, proxies):
    # loop = asyncio.get_event_loop()
    # asyncio.set_event_loop(loop)
    # responses = loop.run_until_complete(send_all_requests(tokens, proxies))
    if platform.system() == 'Windows':
        asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())
    responses = asyncio.run(send_all_requests(tokens, proxies))
    return responses

def multiproc_making(tokens, proxies):
    num_cores = 1 # Our number of CPU cores (including logical cores)
    futures = []
    with concurrent.futures.ProcessPoolExecutor(num_cores) as executor:
        token_start_num = 0
        proxy_start_num = 0
        tokens_len = len(tokens)
        tokens_per_core = math.floor(tokens_len / num_cores)
        proxy_per_core = math.ceil(tokens_per_core / N_PROXY_FOR_REQUESTES)
        for i in range(num_cores - 1):
            new_future = executor.submit(
                start_scraping,
                tokens=tokens[token_start_num:token_start_num + tokens_per_core],
                proxies=proxies[proxy_start_num:proxy_start_num + proxy_per_core]
            )
            futures.append(new_future)
            token_start_num += tokens_per_core
            proxy_start_num += proxy_per_core

```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

```

proxy_for_final_core = math.ceil((tokens_len - token_start_num) / N_PROXY_FOR_REQUESTES)

new_future = executor.submit(
    start_scraping,
    tokens=tokens[token_start_num:tokens_len],
    proxies=proxies[proxy_start_num:proxy_start_num + proxy_for_final_core]
)
futures.append(new_future)
responses = concurrent.futures.wait(futures)

if bool(responses[1]):
    logger.info(f'{responses[1]}')
    logger.info(f'{[response.result() for response in responses[1]]}')
    raise 'Корутина не вернула фьючерс'
all_reponses = []

for response in responses[0]:
    all_reponses += response.result()
return all_reponses

def stopper():
    bot.send_message(chat_id, f'Сталася помилка, програма обробила помилку і продовжила роботу')
    logger.info(f'Сталася помилка, програма обробила помилку і продовжила роботу')
    input()

@logger.catch(onerror=lambda _: stopper())
def main():
    # start1 = time.time()
    tokens = disable_tokens(tokens_data)
    # logger.info(f'Disable tokens time: {time.time() - start1} seconds')
    start2 = time.time()
    responses = multiproc_making(tokens, proxies)
    logger.info(f'Parcing time: {round(time.time() - start2, 4)} seconds')
    start3 = time.time()
    profit_tokens = find_change(responses, tokens)
    logger.info(f'Check profit tokens time: {round(time.time() - start3, 4)} seconds')
    start4 = time.time()
    send_info(profit_tokens)
    logger.info(f'Sending info time: {round(time.time() - start4, 4)} seconds')

# if __name__ == '__main__':
#     logger.info(f'Start parsing...')
#     start = time.time()
#     main()
#     end = time.time() - start
#     print(f'\n\n\n\n\n\n#1 {end} seconds')

if __name__ == '__main__':
    logger.info(f'Start parsing...')
    i = 1
    sum_time = 0
    # while True:
    for _ in range(1):
        start = time.time()
        main()
        end = time.time() - start
        sum_time += end
        avr_time = sum_time / i
        logger.trace(f'#{i} {end} seconds | AVR TIME: {avr_time} seconds')
        print(f'#{i} {end} seconds | AVR TIME: {avr_time} seconds')
        i += 1
    # time.sleep(5)

```

					КПІ.ІП-8426.045430.03.12	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		8

Файл executable_objects2.cpp

```
GLOBAL_TIMEOUT = 10 #Timeout для усіх запитів
N_PROXY_FOR_REQUESTES = 3 #максимальне значення кількості токенів на одиниць проксі
```

Файл async_bitmart_parse

```
import asyncio
import aiohttp
from loguru import logger
import json
import math

#config
from async_config import *
TIMEOUT = aiohttp.ClientTimeout(total=GLOBAL_TIMEOUT)

async def price_by_order(session, symbol, proxy, sum_usdt_for_buy, platforms, slippage):
    url = 'https://api-cloud.bitmart.com/spot/v1/symbols/book'
    params = {
        'symbol': symbol + '_USDT'
    }
    try:
        async with session.get(url, params=params, ssl=False, timeout=TIMEOUT) as response:
            response_text = await response.text()
            try:
                if response.status != 200:
                    raise Exception(f'Expected status code 200, but got {response.status}')
            except Exception:
                logger.info(f'Status code: {response.status}. Bad token: {symbol}')
                logger.info(f'Відправив запит {url} з параметрами {params}. \nОтримав відповідь {response}.')
                logger.info(f'Прокси: {proxy}.')
                logger.debug(f'JSON: {response_text}')
                return 'statusNot200'
            try:
                response_json = json.loads(response_text)
            except json.JSONDecodeError as jsondecodererror:
                logger.info(str(jsondecodererror))
                logger.debug(f'Відправив запит {url} з параметрами {params}. \nОтримав відповідь {response}.')
                logger.debug(f'Text: {response_text}')
                logger.info(f'Прокси: {proxy}.')
                return 'statusNot200'
            buys_orders = response_json['data']['buys']
            amount_usdt = 0
            token_sell_price = None
            total_amount = 0
            for order in buys_orders:
                order_in_usdt = float(order['price']) * float(order['amount'])
                amount_usdt += order_in_usdt
                total_amount += float(order['amount'])
                if amount_usdt > sum_usdt_for_buy:
                    token_sell_price = amount_usdt / total_amount / (1 + slippage)
                    break
            sells_orders = response_json['data']['sells']
            amount_usdt = 0
            token_buy_price = None
            total_amount = 0
            for order in sells_orders:
                order_in_usdt = float(order['price']) * float(order['amount'])
```

					КП.ІП-8426.045430.03.12	Арк.
						9
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

        amount_usdt += order_in_usdt
        total_amount += float(order['amount'])
        if amount_usdt > sum_usdt_for_buy:
            token_buy_price = amount_usdt / total_amount / (1 - slippage)
            break
    answer = {
        'symbol': symbol,
        'token_buy_price': token_buy_price,
        'token_sell_price': token_sell_price,
        'platforms': platforms
    }

    return answer

except (asyncio.TimeoutError, aiohttp.ClientProxyConnectionError, aiohttp.ClientHttpProxyError,
aiohttp.ClientPayloadError,
aiohttp.ClientConnectorError, aiohttp.ClientOSError, aiohttp.ServerDisconnectedError) as err:
    logger.info(f'{err.__class__.__name__}: {str(err)}')
    logger.debug(f'Отправил запрос {url} с параметрами {params}.')
    logger.debug(f'Proxy: {proxy}.')
    return 'statusNot200'

async def get_price_by_order(proxies, tokens, sum_usdt_for_buy=5000):
    async with aiohttp.ClientSession(trust_env=True, timeout=TIMEOUT) as session:
        tasks = []
        n = 0
        i = 0
        for token in tokens:
            if 'bitmart' not in token['exchanges']:
                continue
            if i == N_PROXY_FOR_REQUESTES:
                n += 1
                i = 0
            proxy = f'http://{proxies[n]}'
            task = asyncio.create_task(price_by_order(session, token['symbol'], proxy, sum_usdt_for_buy,
token['platforms'], token['slippage']))
            tasks.append(task)
            i += 1
        answer = {'exchange': 'bitmart'}
        answer['data'] = await asyncio.gather(*tasks)
        return answer

```

Файл async_latoken_parse

```

import asyncio
import aiohttp
from loguru import logger
import json
import math

#config
from async_config import *
TIMEOUT = aiohttp.ClientTimeout(total=GLOBAL_TIMEOUT)

        amount_usdt += order_in_usdt
        total_amount += float(order['quantity'])
        if amount_usdt > sum_usdt_for_buy:
            token_sell_price = amount_usdt / total_amount / (1 + slippage)

```

					КПІ.ІП-8426.045430.03.12	Арк.
						10
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

async def price_by_order(session, symbol, proxy, sum_usdt_for_buy, platforms, slippage):
    url = f'https://api.latoken.com/v2/book/{symbol}/USDT'
    params = {}
    try:
        async with session.get(url, params=params, ssl=False, timeout=TIMEOUT) as response:
            response_text = await response.text()
            try:
                if response.status != 200:
                    raise Exception(f'Expected status code 200, but got {response.status}')
            except Exception:
                logger.info(f'Status code: {response.status}. Bad token: {symbol}')
                logger.info(f'Отправил запрос {url} с параметрами {params}. \nПолучил ответ {response}.')
                logger.info(f'Proxy: {proxy}.')
                logger.debug(f'JSON: {response_text}')
                return 'statusNot200'
            try:
                response_json = json.loads(response_text)
            except json.JSONDecodeError as jsondecodererror:
                logger.info(str(jsondecodererror))
                logger.debug(f'Отправил запрос {url} с параметрами {params}. \nПолучил ответ {response}.')
                logger.debug(f'Text: {response_text}')
                logger.info(f'Proxy: {proxy}.')
                return 'statusNot200'
            buys_orders = response_json['bid']
            amount_usdt = 0
            token_sell_price = None
            total_amount = 0
            for order in buys_orders:
                order_in_usdt = float(order['price']) * float(order['quantity']) # price * amount
                amount_usdt += order_in_usdt
                total_amount += float(order['quantity'])
                if amount_usdt > sum_usdt_for_buy:
                    token_sell_price = amount_usdt / total_amount / (1 + slippage)
                    break
            sells_orders = response_json['ask']
            amount_usdt = 0
            token_buy_price = None
            total_amount = 0
            for order in sells_orders:
                order_in_usdt = float(order['price']) * float(order['quantity']) # price * amount
                amount_usdt += order_in_usdt
                total_amount += float(order['quantity'])
                if amount_usdt > sum_usdt_for_buy:
                    token_buy_price = amount_usdt / total_amount / (1 - slippage)
                    break
            answer = {
                'symbol': symbol,
                'token_buy_price': token_buy_price,
                'token_sell_price': token_sell_price,
                'platforms': platforms
            }
            return answer
    except (asyncio.TimeoutError, aiohttp.ClientProxyConnectionError, aiohttp.ClientHttpProxyError,
            aiohttp.ClientPayloadError,
            aiohttp.ClientConnectorError, aiohttp.ClientOSError, aiohttp.ServerDisconnectedError) as err:
        logger.info(f'{err.__class__.__name__}: {str(err)}')
        logger.debug(f'Отправил запрос {url} с параметрами {params}.')
        logger.debug(f'Proxy: {proxy}.')
        return 'statusNot200'

async def get_price_by_order(proxies, tokens, sum_usdt_for_buy=5000):
    async with aiohttp.ClientSession(trust_env=True, timeout=TIMEOUT) as session:

```

					КП.ІП-8426.045430.03.12	Арк.
						11
Змін.	Арк.	№ докум.	Підп.	Дата.		

```

tasks = []
n = 0
i = 0
for token in tokens:
    if 'latoken' not in token['exchanges']:
        continue
    if i == N_PROXY_FOR_REQUESTES:
        n += 1
        i = 0
    proxy = f'http://{proxies[n]}'
    task = asyncio.create_task(price_by_order(session, token['symbol'], proxy, sum_usdt_for_buy,
token['platforms'], token['slippage']))
    tasks.append(task)
    i += 1
answer = {'exchange': 'latoken'}
answer['data'] = await asyncio.gather(*tasks)
return answer

#
# tokens = [{
#     "symbol": "BTC",
#     "platforms": {
#         "ethereum": "0xbc7250c8c3eca1dfc1728620af835fca489bdf3"
#     },
#     "exchanges": [
#         "latoken"
#     ],
#     "slippage": 0,
#     "coingecko_id": "btc"
# }, {
#     "symbol": "ETH",
#     "platforms": {
#         "ethereum": "0xbc7250c8c3eca1dfc1728620af835fca489bdf3"
#     },
#     "exchanges": [
#         "latoken"
#     ],
#     "slippage": 0,
#     "coingecko_id": "eth"
# }]
#
# with open('proxies-http-host.txt', 'r', encoding='utf-8') as f:
#     proxies = [line.strip() for line in f]
#
# async def send_all_requests(tokens, proxies):
#     tasks = []
#     tasks.append(asyncio.create_task(get_price_by_order(proxies, tokens)))
#     return await asyncio.gather(*tasks)
#
# asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())
# responses = asyncio.run(send_all_requests(tokens, proxies))
# pprint(responses)

```


Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для аналізу даних при купівлі-продажу
криптовалют на ринку бірж
Програма та методика тестування
КПІ.IT-8426.045430.04.51**

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Максим СОЛОВЕЙ

Київ – 2023

ЗМІСТ

1 Об'єкт випробувань	3
2 Мета тестування	4
3 Методи тестування.....	5
4 Засоби та порядок тестування.....	6

					КПІ.ІТ-8426.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		2

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є парсер криптовалют.

					КПІ.ІТ-8426.045430.04.51	Арк.
						3
Змін.	Арк.	№ докум.	Підп.	Дата.		

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- Перевірка якості роботи та валідація результатів;
- Підрахунок бізнес метрик;

					КПІ.ІТ-8426.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- Юзабіліті-тестування. Його мета - виявити слабкі місця додатку за допомогою спостереження за обраними людьми (персонами), які користуються додатком. Після цього відбувається покращення додатку на основі знайдених слабких місць.
- Час надання. Перевірка часу на виконання запитів.

					КПІ.ІТ-8426.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Додаток тестують три особи. Кожен з них представляє тип людини (персони), яка може користуватися сайтом. Перша особа – це експерт (з псевдонімом: Експерт), який розуміється на екосистемі криптовалют та криптовалютних бірж. Другий - молода людина, яка не дуже добре знає криптовалюту, але в цілому відноситься до типу ранніх послідовників (з псевдонімом Adopter). Третій - людина середнього віку (з псевдонімом Middle-aged), яка регулярно працює з комп'ютером, але виключно на рівні на рівні користувача. Тестування ґрунтується на спостереженні за вже згаданими особами по черзі одну за одною. Спочатку криптовалютний арбітраж та криптовалютні біржі коротко пояснюються криптовалютний арбітраж та криптовалютні біржі. Потім персонажам дають список завдань, а потім спостерігають за тим, як вони з ними справляються. спостерігають за тим, як вони справляються із завданнями. Після завершення сесії запитують про їхнє загальне враження від програми.

Бізнес метрики криптопарсера. Бізнес-метрики парсеру криптовалют можуть відрізнятися залежно від конкретних цілей і завдань бізнесу. Однак, ось деякі загальні бізнес-метрики, які можуть бути релевантними для парсеру криптовалют:

- Кількість підтримуваних криптовалют. Ця метрика відображає загальну кількість криптовалют, які підтримує парсер. Для парсеру важливо охопити широкий спектр популярних криптовалют, щоб залучити користувачів і надати вичерпні ринкові дані.

- Точність даних. Ця метрика вимірює точність аналізованих даних. Для парсеру криптовалют дуже важливо надавати надійну і точну інформацію, включаючи ціни в реальному часі, обсяги торгів, ринкову капіталізацію та історичні дані. Точність даних забезпечує довіру і надійність, а також впливає на задоволеність і довіру користувачів.

					КПІ.ІТ-8426.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		6

– Швидкість парсингу - це час, необхідний парсеру для отримання, обробки та оновлення криптовалютних даних. Швидкість парсингу дозволяє користувачам швидко отримувати доступ до даних в режимі реального часу, що особливо важливо на швидкозмінному ринку. Оптимізація швидкості парсингу може покращити користувацький досвід і залучити більше користувачів.

– Час роботи API - це метрика, яка вимірює доступність і надійність API парсеру. Вона відображає відсоток часу, протягом якого API парсеру доступний і функціонує. Вищий час роботи API свідчить про кращу надійність, мінімізацію перебоїв у доступі до даних для користувачів і забезпечення безперервного обслуговування.

– Залучення користувачів. Показники залучення користувачів включають такі фактори, як кількість активних користувачів, коефіцієнт утримання користувачів та активність користувачів на платформі парсеру. Ці показники дають уявлення про те, як часто користувачі взаємодіють з парсером, як довго вони залишаються залученими і чи знаходять вони цінність у наданих даних.

					КПІ.IT-8426.045430.04.51	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		7

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

**Програмне забезпечення для аналізу даних при купівлі-продажу
криптовалют на ринку бірж
Керівництво користувача
КПІ.IT-8426.045430.05.34**

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Максим СОЛОВЕЙ

Київ – 2023

ЗМІСТ

1 Загальні відомості	3
2 Підготовка до роботи.....	4
2.1 Системні вимоги для коректної роботи	4
2.2 Завантаження застосунку.....	4
2.3 Перевірка коректної роботи.....	4
3 Робота із програмним забезпеченням.....	5

					КПІ.ІТ-8426.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		2

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмне забезпечення для аналізу криптовалют демонструє здатність отримувати дані в режимі реального часу з криптовалютних бірж, блокчейн-мереж та інших відповідних джерел. Воно включає в себе передові алгоритми і методи для вилучення та організації ключової інформації, такої як ціни, обсяги торгів, деталі транзакцій і ринкові тенденції. Ці дані можуть надати цінну інформацію для трейдерів, інвесторів та дослідників, дозволяючи їм приймати обґрунтовані рішення в динамічному світі криптовалют.

Цей проект розроблений мовою програмування Python 3. В цілому, розробка програмного забезпечення для парсеру криптовалют на Python має потенціал для революційної зміни способу взаємодії приватних осіб та організацій з криптовалютними даними. Поєднуючи потужність Python з ефективним тестуванням системи, ми можемо створити надійний і ефективний інструмент, який дає користувачам можливість орієнтуватися в складному і постійно мінливому світі криптовалют.

					КП.ІТ-8426.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		3

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність ПК з операційною системою на базі Windows/Ubuntu/macOS;
- встановлена остання версія Python ≥ 3.9 ;
- для запуску ПЗ на пристрої має бути не менше 16 GiB;
- для завантаження даних, зберігання оброблених даних, зберігання навчених моделей, рекомендується мати не менше 6 GiB пам'яті на жорсткому диску.

2.2 Завантаження застосунку

Застосунок можна розгорнути встановивши всі необхідні залежності із файлу requirements.txt шляхом виконання команди:

```
pip install -r requirements.txt
```

Для запуску ПЗ потрібно запустити файл main.py:

```
python main.py
```

2.3 Перевірка коректної роботи

Після запуску програми можна побачити індикатори прогресу відбуваючих процесів роботи ПЗ. В кінці виконання ПЗ буде виведене повідомлення про фінальні результати та результати тестування.

					КПІ.IT-8426.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		4

3 РОБОТА ІЗ ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

Після першого запуску програмного забезпечення, з'являється можливість запускання лише окремих його модулів наразі із перевикористанням результатів минулих запусків. Результати минулих запусків зберігаються в форматі rqt, тому в будь-який момент часу їх можна використати повторно, тим самим зменшити час виконання ПЗ.

Конфігурація запуску окремих компонент проводиться за допомогою файлу налаштувань – settings.py. Наведемо опис доступних налаштувань:

- FORCE_PREPROCESS – запуск модуля попередньої обробки даних;
- FORCE_RETRIEVAL – запуск модуля відбору кандидатів на рекомендацію;
- WEEK_NUM – довжина проміжку часу для пошуку кандидатів в історичних даних (збільшення цього параметру призводить до кращої якості фінального результату, проте збільшує вимоги до об'єму оперативної пам'яті комп'ютера)
- VALIDATE – проведення валідації результатів

При першому запуску файл конфігурації ігнорується.

Результати модулів обробки даних та відбору кандидатів зберігаються в директорії dataset. Моделі ранжування знаходяться у директорії models.

					КПІ.IT-8426.045430.05.34	Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.		5

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2023 р.

Програмне забезпечення для аналізу даних при купівлі-продажу

криптовалют на ринку бірж

Графічний матеріал

КПІ.IT-8426.045430.05.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олег ЛІСОВИЧЕНКО

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Максим СОЛОВЕЙ

Київ – 2023